Computer Aided Diagnosis of Miliary TB in Chest X-Rays

Anthony Koeslag

Submitted to the Department of Electrical Engineering,
University of Cape Town, in fulfilment of the requirements
for the degree of Master of Science in Engineering.
November 2002

Declaration

I declare that this dissertation is my own, unaided work. It is being submitted for the degree of Master of Science in Engineering at the University of Cape Town. It has not been submitted before for any degree or examination at this or any other university.

Signed by candidate

.....

Anthony Koeslag

(Candidate's Signature)

Acknowledgements

I would like to thank the following for their contribution towards this thesis:

- Professor Gerhard de Jager, my supervisor, for his enthusiasm and guidance.
- African Medical Imaging and UCT for financial support.
- The members of the Digital Image Processing Group for their support and contributions.
- Prof. Steve Beningfield for his valuable assistance with the medical aspects of this project.

Abstract

With the improvement in computer technology, Computer Aided Diagnosis (CAD) is becoming an increasingly more powerful tool for radiologists. The focus of this project was on CAD of pulmonary miliary tuberculosis. Several methods for enhancing lung textures were discussed as an aid to the radiologist in diagnosing miliary TB. Some statistical approaches and template matching methods were used to measure characteristics of both healthy and unhealthy (miliary TB) lung textures. These measurements were evaluated to see if a computer can be programmed to differentiate between lung texture from a healthy lung and lung texture from a lung with miliary TB.

Contents

List of Figures

Chapter 1

Introduction

Conrad Wilhelm Rontgen's discovery of X-rays and their properties [39, 13], in 1895, made it possible for physicians to examine internal organs of their patients without the trauma of surgery. This discovery revolutionized medical diagnosis as we know it today and lead to him being the first person to receive the Nobel prize, in 1901. Rontgen's discovery in fact made him the very first person to do medical imagery, a field which is still developing at an impressive rate as new applications and methods are being devised.

Despite many types of examinations being replaced by Computer aided Tomography (CT) and Magnetic Resonance Imaging (MRI) scans, chest X-rays still remain the most common form of radiology [40]. This is due to the vast amount of information that can be learned about a patient's health from the chest X-ray. Many forms of lung disease can be diagnosed from examination of the lung images. Tuberculosis is one of the many diseases that is primarily diagnosed through the use of chest X-rays.

Tuberculosis (TB) has been the main cause of death in many countries, and indeed world wide [14]. The disease primarily affects the lungs, where it can cause severe damage and scarring. The bacterium causing the infection, tubercle bacillus, can be identified in the sputum of an infected patient, however this is only possible after the infection has caused open ulcers in the lungs. At this point the disease it very contagious as it can become airborne when the patient coughs. Louis Pasteur

discovered that micro-organisms caused disease and fermentation in the mid 19th century [12]. Later Robert Koch's discovery that consumption, now known as Tuberculosis, was in fact caused by a bacterium [11]. These important discoveries meant that most cases of TB could be cured, provided they were diagnosed early enough. A chest X-ray remains the best method for diagnosing the disease in its early stages. 93% of all recognized cases of TB are diagnosed with a chest examination [40].

Pasteur and Koch's contribution to medicine resulted in the decline of the rate of TB cases from the mid 19th century until fairly recently. In the mid 1980's there was a dramatic change. Most African countries south of the Sahara have had extremely high rates, and even in the United States, the United Kingdom, and some other first world countries, the number of TB cases each year has been increasing [30]. This is primarily attributed to the spread of HIV.

Early diagnosis is critical for fast and effective treatment but in their early stages, most diseases can be very difficult to detect. Even a highly skilled radiologist will make mistakes, or overlook vital, but faint, detail, resulting in an incorrect diagnosis. Furthermore, people performing highly repetitive tasks, for example doing a mass screening for TB, often loose concentration and begin to make mistakes at an increasing rate.

Krupinski [7] found that having a second radiologist re-evaluate images decreased the number of incorrect classifications. This solution is, however, cost-intensive as it requires an additional human assessment of each patient by a highly qualified professional.

Since the industrial revolution there has been a strong drive to create machines that could do tasks typically associated with people more effectively. Although this caused mass unemployment in some cases, there are also cases where the benefits far outweighed the negatives. Why, for instance, endanger people by sending them into highly dangerous situations, when a robot can do the same job as well, if not better. Thus, in 1967 a robotic lunar lander, called *Surveyer III*, landed on the moon. This provided a safe method for sampling the surface of the moon and in turn the information gained, made it safer for the 1969, manned, Apollo 11 moon landing.

More generally, however, the industrial revolution freed people from doing boring, highly repetitive tasks. The advantages have been, not only that persons have been able to concentrate on more creative activities, but that mundane and repetitive tasks could be performed tirelessly and with extreme accuracy by mindless machines.

Thus machines have improved the working conditions and efficiency of certain endeavours. Such machines should, therefore, be regarded as tools. Similar tools, specifically designed to perform routine diagnoses, can be used in radiology; allowing the radiologist to concentrate on more complex problems [7].

Radiology is a field that has seen many advances in the tools it uses. In 1917 J. Radon proved that three dimensional objects can be created from an infinite set of two dimensional projections [32]. This was the start of tomography, which allows the radiologist to make an image of a slice through a patient. Since then computers have been used to improve the quality and speed of tomography. The resulting images are called computer aided tomography (CAT), or computer tomography (CT), scans.

With the constant advancement of computer technology, there has been an increasing integration of computers into the field of radiology. Apart from computers being used to acquire and reconstruct images, many tools have been, and still are being, developed to aid the radiologist. This general and varied field of applications is referred to as computer aided diagnosis (CAD). The Three main areas of CAD research fall into the following categories: enhancement, segmentation and analysis.

Enhancement of images or details of images is specifically done to make objects easier to see in an image. The principle problem is the difficulty in evaluating the effectiveness of enhancing an image. Measuring the image quality can be used to evaluate some image enhancement algorithms. Henrichsen [2] looks at edge effects as a measure of image quality. Bernardini and Kovacevic [31] measure the amount of structured noise in an image to determine the quality of an image. Image quality also depends on the the acquisition and storage of images. Okumura et al [3] investigate the effects of directly digitizing microscopic images. They noted that microscopic images had a different spatial frequency than regular images, and

therefore got distorted more by image compression algorithms such as JPEG.

Suzuki et al [16] investigated contrast mapping to improve diagnostic quality of images displayed on CRT monitors. Although the images were enhanced, the luminescence of CRT displays is considerably less than that of the light boxes used to view X-ray films. This does not effect the viewing of bones, but to view dark regions it is necessary to make the room much darker. They also noted that other enhancement processes held major prospects for improving the diagnostic performance of digital images.

To overcome the inherent difficulties of evaluating the results of image enhancement, Cagnoni and Poli [34] used an interactive genetic algorithm to evolve pseudo colour enhancement algorithms. They found that the user input improved the efficiency of the genetic algorithm. Poli [33] also used genetic algorithms for enhancement, feature detection and image segmentation by considering each task as a image filtering problem. By setting criteria for the fitness functions, he was able to produce results which outperformed neural networks.

Enhancement can also be used as a preprocessing tool to help segmentation algorithms. Li et al [9] used wavelet analysis to enhance and segment misidentifications in digital mammograms. As breast cancer accounts for more deaths in females, than any other cancer, it has seen a significant amount of research [25]. Early diagnosis can dramatically improve results in the treatment of this form of cancer.

Segmentation is used to define specific regions in images, this is often coupled with measurement algorithms. Koeslag [4] used a simple threshold to segment and measure the dilation of the ventricles of the brain. Dian [6] used adaptive thresholds in a knowledge based system to segment different parts of the body in a full body X-ray image. Reinhardt et al [23] used a maximum-likelihood and model based method to estimate the radius of airways in chest CT images. This method takes into account the blurring of edges in X-ray images, which improves the accuracy of the measurements over the half-max method. Automation of this process significantly increases the speed and accuracy of the measurements. There have been investigations of methods for the automatic detection of lung nodules, for the early detection of lung cancer [10].

Measurement algorithms are used to measure objects and details in images. Often the measurements are traditionally done by hand, but the accuracy and speed of these measurements can be improved by automating them. Computers can also be used to make measurements that the human eye can not perform effectively, for example quantitative texture analysis. Vincent [24] used granulometry to measure lung texture in the CAD of silicosis, a lung disease often found in coal miners. Scott [29] used image registration and subtraction to measure the symmetry in medical images.

Image registration is used to warp two similar images so that mutual information in the images are in the same locations in both images. Declerck et al [15] used image registration to measure the orientation of the heart in single photon emission CT images. Automating this process decreased the variation in measurements caused by the operator. Zhang and Huang [17] used a statistical measure of intensity distribution and edge orientation to automatically recognize and remove the background from X-ray images.

In this project several methods were investigated to determine their value for computer aided diagnosis of miliary TB. Image enhancement algorithms were investigated to determine their effectiveness at enhancing textures in the lungs. Effective enhancement of lung textures may make the early diagnosis of lung diseases easier for the radiologist. Algorithms for analyzing lung textures were evaluated to determine if a computer could be used in a mass screening for miliary TB. Such a system would remove some of the human error inherent in such routine diagnosis. The diagnosis of miliary TB is a relatively simple task for a radiologist to perform. If a computer can be programmed to perform this simple task, this process can be extended to solve other more sophisticated problems. General computer aided diagnosis is to broad a problem to be solved at this time. This might become possible as more of these relatively simple problems become solved.

Chapter 2

The Data Set

It is important to understand the properties of a data set in order to be able to effectively process the information. In an image processing project, like this one, everything from the manner in which the images are acquired, developed, scanned into a computer and stored has profound effects on the information in the images [18].

2.1 X-rays used in Radiology

A huge step in medical diagnosis was the discovery of X-rays, and the invention of the machines necessary for using them to make X-ray images of people. This allowed physicians to be able to look 'inside' a patient without surgical procedures. X-rays pass easily through less dense objects, such as air and water, but get more readily absorbed by denser objects like bone and metal. Because of this, X-ray images are lightest where fewer X-rays have passed through the object, and darker where more X-rays have passed through the object to hit the photographic plate. The ability of X-rays to penetrate an object is dependent on the frequency of the X-rays. The higher the frequency the more transparent an object is to X-rays.

When X-rays encounter a dense object, some rays pass through the object, some are absorbed, but some are also scattered. This scattering causes a blurring around

the object in the developed image. This blurring is significantly reduced by putting a lead grid in front of the photographic plate, so that only X-rays travelling parallel to the holes in the grid, will affect the final image. However this means that the X-ray dosage has to be increased significantly in order to get enough X-rays to hit the photographic plate. This grid can sometimes be seen in the image. A fine vertical pattern can be seen in Figure 2.1 resulting from the lead grid used to filter the X-rays after passing through the patient.

Figure 2.1: A fine vertical pattern resulting from the lead grid used to filter the X-rays before they hit the photographic plate.

Wavelength and penetration is dependent on the generating voltage and can be varied depending on the application. X-rays generated by low voltages have longer wavelengths and penetrate objects to a lesser extent than X-rays generated using

a higher voltage. In radiological applications the lowest voltages are used for applications concerning soft tissue. A typical application using low voltages is mammography. Mammography is done in the range of 25 to 45 kilovolts. Most examinations, including chest X-rays, are done in the 100 to 125 kilovolt range.

Voltage is not the only factor in taking an X-ray. The intensity of the radiation can be varied by changing the current between the anode and cathode of the X-ray tube. It is also absolutely important that the patient remain absolutely motionless during the exposure. While every effort should be made to reduce patient motion to a minimum, it cannot be eliminated completely. The patient can be instructed to hold their breath, but the heart will continue to pump and cause a blurring. Duration of the exposure is therefor also a factor. The shorter the exposure, the higher the amperage used. Typically the current used in diagnostic work is in the 200 to 500 milliamp range.

X-rays taken at different settings, of voltage and amperage, produce different looking images. Unfortunately the settings used to take a chest X-ray depend largely on the machine and the X-ray source being used.

The developing of the images is similar to photographic techniques and is carried out in a dark room. The film used is typically a silver halide photographic emulsion on a cellulose acetate base, generally referred to as silver acetate. Over the years the sensitivity and durability of this film has been improved. This development has been partly a result of the developing technique changing. Developing techniques have also been automated, but evidence of people handling X-rays is still present in the images. Figure 2.2 shows a number of marks and fingerprints in a chest X-ray. These marks were caused during or soon after the images were developed.

The images in the data set for this project were taken from the Groote Schuur X-ray library. The images in this library were arranged in different classes depending on the features in the image. Some of the images in these classes came from other hospitals and date back as far as 1960. These images have a number of defects that have a negative effect on a computer aided diagnosis algorithm. The defects present in this data set were varied, and included: Fingerprints (see Figure 2.2), scratches (see Figure 2.3), smears (see Figure 2.4) and writing on the image (see

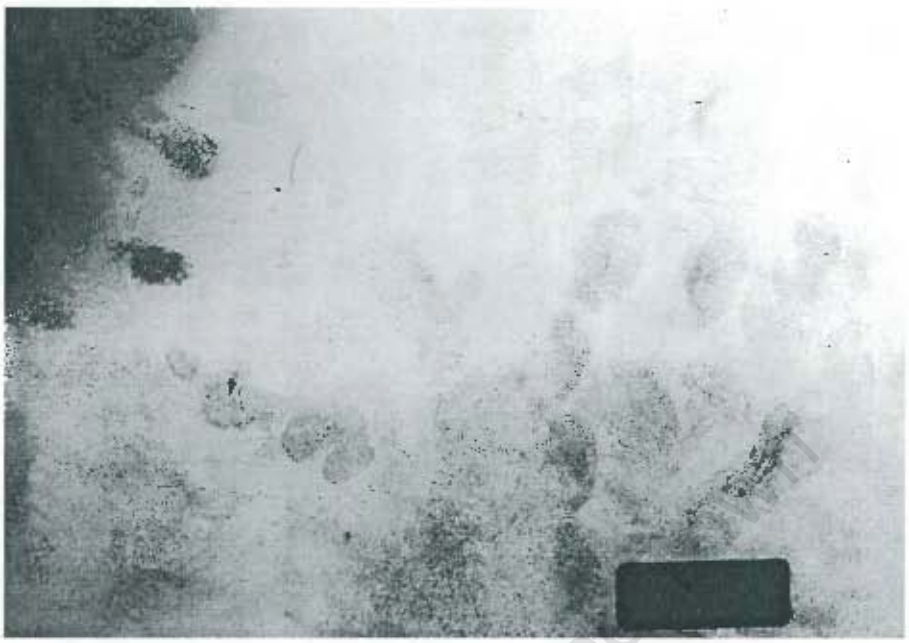

Figure 2.2: A number of marks and fingerprints on an X-ray. These marks were caused either during the development process or shortly after it.

Figure 2.5). These defects were all caused after the image had been developed.

The contrast of the image is affected by the voltage and current settings of the X-ray machine and can also be altered during the developing process. The contrasts in Figure 2.6 were such that the edges of the lungs could not be clearly defined. The contrast in the dark region of Figure 2.7 (shown by the arrow) was very low in comparison to the rest of the image. The texture of the lungs in this section could not be effectively analyzed.

Other artifacts in the images were caused even before the X-ray was taken. Figure 2.8 shows a number of artifacts caused by electrodes on the patients chest. The location and type of artifacts that are caused by objects on or in the patients body are unpredictable. As a result these artifacts cause problems when a computer programme tries to analyze the images.

X-ray plates have to be digitized before they can be processed. These plates can

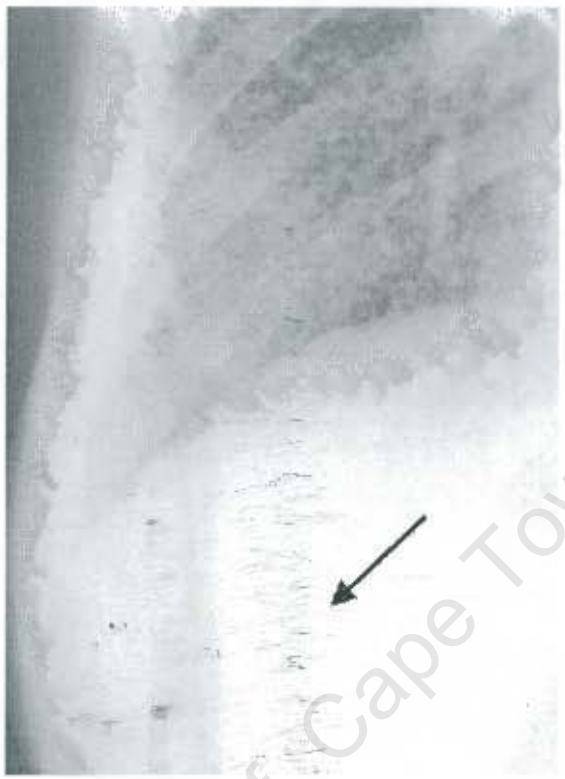

Figure 2.3: Scratch marks were caused in a variety manners, and the older the image is the more likely that the surface of the image has been damaged.

not be scanned by conventional scanners, as the acetate is very reflective. To avoid this problem, the scanners built to scan X-rays use lasers instead of the conventional light sources. A Lumysis scanner was used to scan the images in this data set. Artifacts were also caused by mechanical problems with the scanner. Figure 2.9 shows artifacts caused by a problem with the rotation of the mirror used to deflect the laser beam during scanning. The images with these artifacts were useless, but often re-scanning solved the problem.

Finally, artifacts can also be caused by the compression algorithms used to store the images once they are in a computer. For this project the data set was stored as 14 bit raw images. As the raw image has no compression there was no loss of

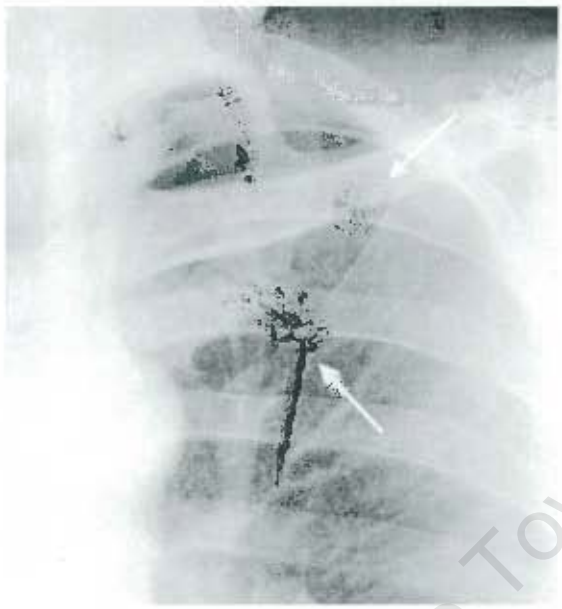

Figure 2.4: These sorts of marks were caused during the developing process.

information.

2.2 Image Classification

As the images in the library were used as a learning aid for medical students, often the cases had more than just one pathology present. Also many of the case studies showed a series of images of the same patient as the disease progressed, and so often there were healthy images in a case study of a person with miliary TB. Sometimes a person with miliary TB had also acquired a secondary disease. Some of these secondary diseases obscure the effects of the original disease, which made computer unable to correctly process these images.

As a result of these problems, it was necessary to have the data set correctly classified. A small, 300 by 300 pixel block, was selected from each of the images in the data set. These selections were chosen so that only lung texture, and the

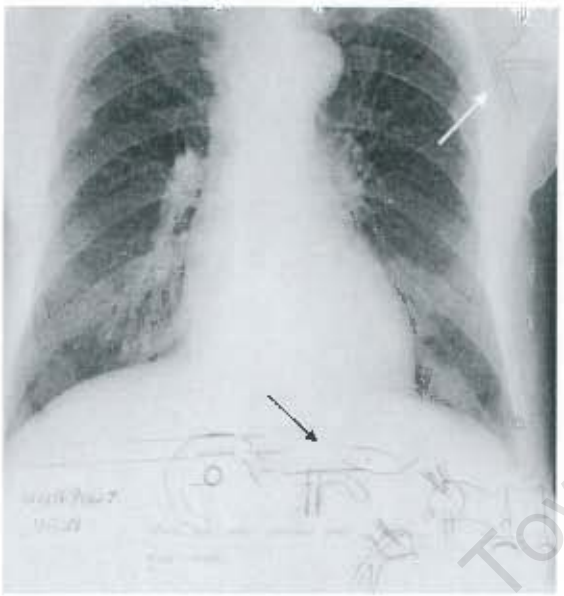

Figure 2.5: As these images came from a library, that is used as a learning aid for students, some of the images had notes written on them. This caused problems when writing obscured a region of interest.

occasional rib, was visible. It was necessary to limit the contents of each images to the texture of the lungs, as this was all that the computer would be given to classify an image, and the radiologist might have been able to classify an image by another feature not present in the texture of the lungs. These textures were taken to a radiologist for classification. Images that were part of the miliary TB set that were classified as healthy, or as having another disease, were removed from the data set.

As a result, the healthy set had 55 images, and the miliary TB set had 28 images in it.

Figure 2.6: An image with bad contrast. In many areas the edges of the lungs were not clearly defined. The details of the lung textures were also less distinct.

Figure 2.7: A section of lung where the contrast was such that in some sections the of the lungs (marked by the arrow) there was no detail.

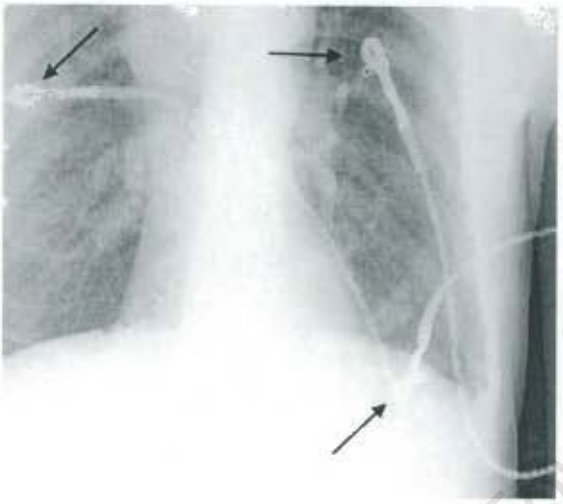

Figure 2.8: A patient may have an assortment of objects in or on their bodies. These wires were in fact electrodes that had been placed on the patients skin.

Figure 2.9: This image shows a number of artifacts caused by a mechanical fault during the scanning process.

Chapter 3

Review of Image Enhancement

Computer Aided Diagnosis of X-ray images comprises two broad categories of techniques. They are Image Enhancement and Automatic Detection. Image Enhancement helps the radiologist to see information that might otherwise be overlooked. Making the diagnosis remains the radiologist's province. The computer does not suggest any diagnoses; it is merely used to present the information more clearly. The problem, however, is that it might enhance irrelevant information to the point that it obscures important information. In Automatic Detection, the subject of this thesis, the aim is to perform some of the radiologist's diagnostic tasks.

This chapter briefly describes several image enhancing techniques that can be used on chest X-rays, and highlights the differences between them.

3.1 The Human Visual System

The human visual system (HVS) comprises the eyes and optic centres of the brain. Together they constitute the tools by which we perceive the world we live in. It is necessary to have some understanding of the strengths and weaknesses of this system in order to enhance images effectively. As the HVS is particularly sensitive to edges between different colours, contrasting tones and varying textures [8, 26],

image enhancement of chest X-rays is largely aimed at contrast enhancement. To illustrate this sensitivity of the HVS to edges, Figure 3.1 consists simply of a number of lines which the HVS instantly interprets as an ant. In reality the squiggly lines on the white background are totally unlike any biological, 3d, full colour ant. The visual system extracts the outlines of the biological ant, and immediately recognizes the close correspondence between this outline, and the black line trace in Figure 3.1.

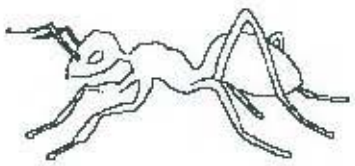

Figure 3.1: A squiggly line which the HVS immediately interprets as, not just the outline of an ant, but as 'an ant'.

Though preoccupied with the detection of edges, the HVS is capable of distinguishing only between approximately 40 different levels of light in a single section of an image [39, 26]. This can make the finding of an object whose shape is defined by subtle contrasts very difficult to do with the naked eye. Computers can be used to enhance these subtle contrasts in images, thus making it easier to detect these faint but potentially important objects.

3.2 Edge Detection

As the HVS is predominantly preoccupied with edges, a method for detecting edges has been investigated. This method was chosen because it was very simple, and it left the location and width of the edges undistorted. Edge detection is, however, extremely sensitive to noise [21], and so some of the features of interest became unrecognizable due to fortuitous, random, or irrelevant edges.

Edge enhancement results in an image with bright pixels for sharp edges, darker pixels for soft edges, and black where no edges were detected. The resulting image is referred to as an 'edge image' as it contains only edge information.

Edges are measured as differences between neighboring pixel values. Thus two adjacent pixels that have a large difference in values are seen as having a sharp edge, which is coded as a bright pixel in the edge image. Where adjacent pixel values differ less, the edge is said to be softer and is coded as a less bright pixel in the edge image. This method is implemented by systematically examining each pixel and its neighbor, and placing the absolute value of the difference between the two pixels into a pixel in the edge image located at the site of the first pixel. Using this process it is possible to look at differences between neighboring pixels in the horizontal, vertical and diagonal directions. As the edges in the chest X-rays are not direction specific, the sum of the differences between horizontal and vertical neighbors was used in the examples which follow.

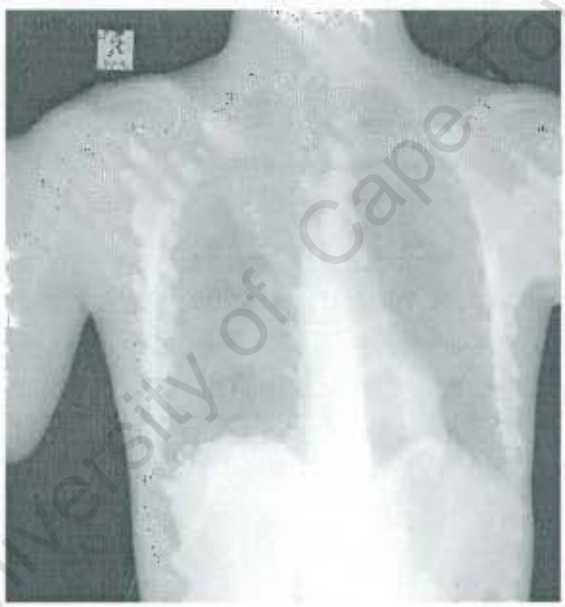

Figure 3.2: An unprocessed chest X-ray

Figure 3.3 shows the edge image of Figure 3.2. The tag in the top left corner of the image had very sharp edges compared with the rest of the image, and therefore stands out disproportionately. Figure 3.4 shows the same edge image after windowing. This made all the soft edges more distinct. However some other artifacts also became visible: six horizontal lines appeared in the picture. These initially

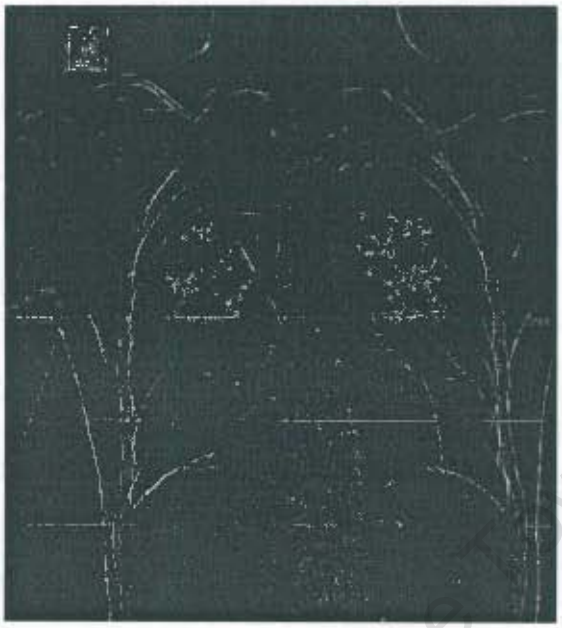

Figure 3.3: The edge image from a chest X-ray before windowing

invisible lines were caused by the overlapping of the CCD cameras used in the acquisition process. This was however an early Lodox image, and artifacts such as these have been largely eliminated in more recent images. The edge enhancement technique also made the background noise more prominent.

The Lodox machine, used to scan this figure 3.2, is a low dosage X-ray machine that produces a digitized image on a computer rather than an acetate plate. This removes many of the problems associated with developing and then scanning the conventional X-ray plates. The Lodox machine is also capable of scanning the full length and width of a person, thus reducing the number of exposures required to X-ray a patient.

The edge detection greatly enhanced bone structures in the chest X-ray. This is particularly well illustrated in Figure 3.5 which is a close up of the right shoulder. Other objects that were too faint to see have became distinct. The t-shirt that this patient was wearing became visible (Figure 3.6), the collar of which can be easily seen in Figure 3.5. These artifacts could have caused problems if they overlapped

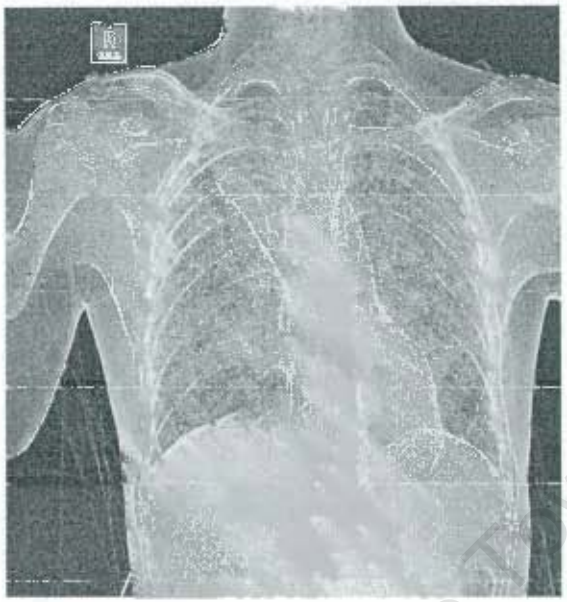

Figure 3.4: The edge image from a chest X-ray after windowing

the patient's body and became incorporated into the lung texture. Although a great deal of useful information was enhanced, the blood vessels and air passages in the lungs, (Figure 3.7) which were the main concern of this project, became indistinguishable from each other. Over-distorting lung texture in the edge images can therefore sometimes be counter-productive.

3.3 Homomorphic Image processing

The term homomorphic refers to a system that has an algebraically linear transform between input and output, and therefore should be reversible [5, 22]. This is an old term, in which low and high spatial frequencies in an image are referred to as luminescence and reflectance respectively. This method is simply a high pass filter implemented in the Fourier domain, which reduces the dynamic range and increases the contrast of the image in the space domain. It has the advantage over the edge detection in that low frequencies can be partially suppressed instead of

Figure 3.5: A closer look at the edge information in the shoulder

being removed. Thus enhancing edges instead of only detecting them. This means that the blood vessels remained lighter than the air passages on a chest X-ray picture, but their edges were still enhanced, which made them more conspicuous.

An example of the Fourier domain magnitude components of an image can be seen in Figure 3.8. The filter must be converted into two dimensions. This is done by creating a two dimensional matrix that has the shape of the filter in all directions (radialy) from each of the corners, as seen in Figure 3.9. Figure 3.10 shows the result of filtering Figure 3.2. The filter suppressed the low frequency by 50% and rose to unity over a distance of 5% of the width of the Fourier domain.

This method can easily perform multiscale [37, 1] contrast enhancement. This means it can enhance different spatial frequencies in the image by different amounts. The problem, however, is that any differential adjustment of spatial frequencies in the image will affect the entire image.

It is very difficult to objectively grade the results of detail enhancement [2, 31]. As the focus of this project was on the recognition of miliary TB which alters the

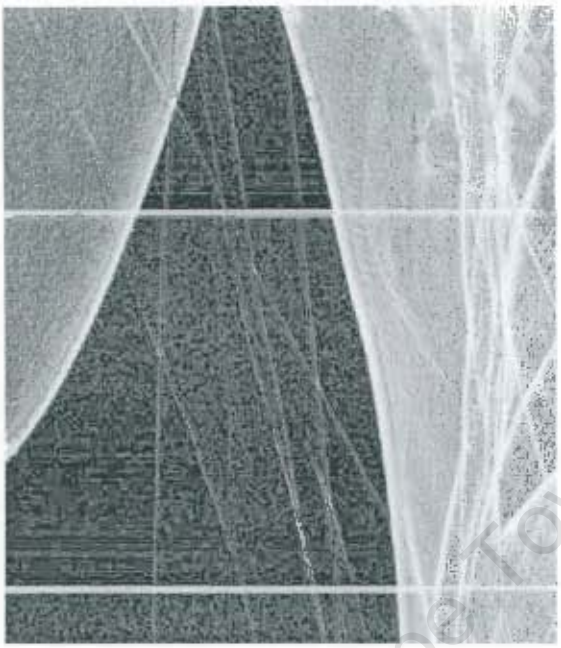

Figure 3.6: A closer look at the edge information showing the t-shirt and some camera artifacts

appearance of the lung texture, one obvious criterion for the detail enhancement was that it should enhance the detail in the lungs. Another criterion was that the rest of the image should be distorted as little as possible. This was achieved by creating an image using an arbitrarily chosen filter, analyzing the resulting image and then adjusting the filter slightly, at which point the process was repeated until the desired result (or as close to it as possible) was achieved. The desired result being the enhancement of the lung textures with as little distortion as possible.

3.4 Unsharp Masking

Earlier in this chapter a method of edge detection was discussed. Unsharp masking is another method of edge detection [18, 25]. This method creates a blurred, or low pass, version of an image which is then subtracted from the original image.

Figure 3.7: A closer look at the edge information of the lungs. The edges of the pulmonary blood vessels and bronchi have created an uninterpretable mess of squiggles

Figure 3.11 demonstrates how edges are detected using this method in a simple example. Notice that the detected edges are wider than they were in the input signal. This effect is directly related to the size and shape of the blurring function. Variation in effects, achieved by choosing different blurring functions, gives this method its versatility.

This method is also used to enhance edges (rather than just detecting them, as illustrated in Figure 3.11) by multiplying the blurred image by a number between 1 and 0 before subtraction. This leaves a portion of the low frequency information in the resulting image. Figure 3.12 shows how this technique was used to enhance the detail in Figure 3.2. This method introduced considerably less distortion than the homomorphic image enhancement.

Multiscale enhancement is achieved by subtracting various blurred images, each

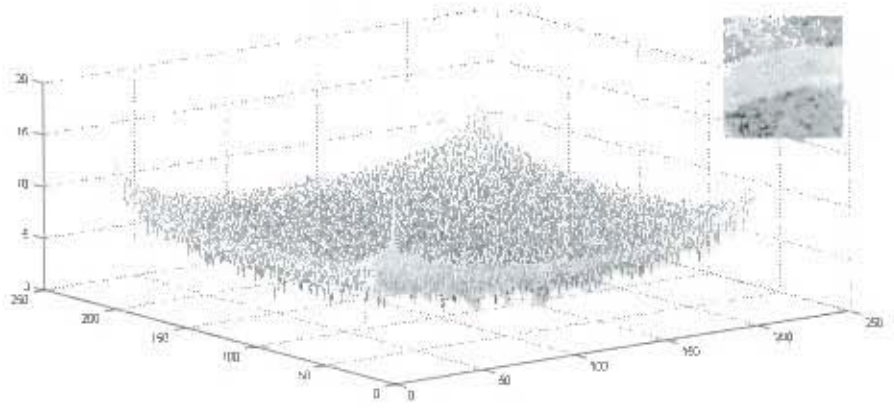

Figure 3.8: The magnitude of the Fourier domain components of a section of lung texture. The lung texture can be seen in the top right corner.

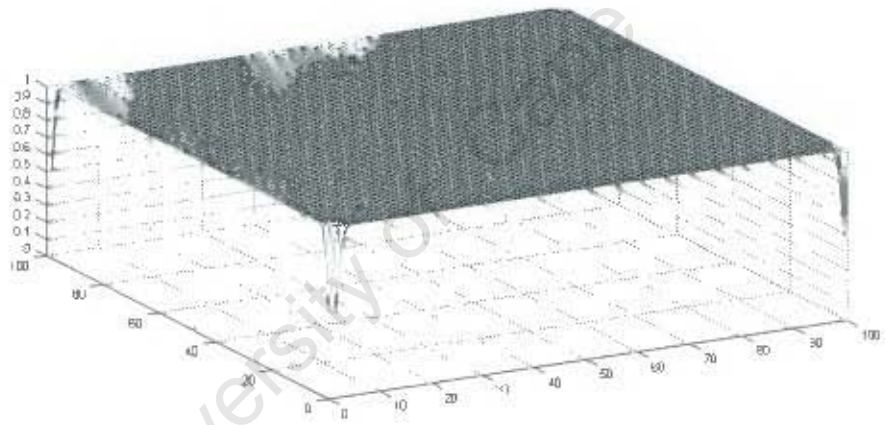

Figure 3.9: A two dimensional Fourier domain filter.

blurred by a different size blurring function. Each blurred image can be multiplied by a different number before subtraction depending on the desired effect. This process becomes extremely computationally intensive, and is therefore very slow, as well as requiring large amounts of memory and disk space to complete.

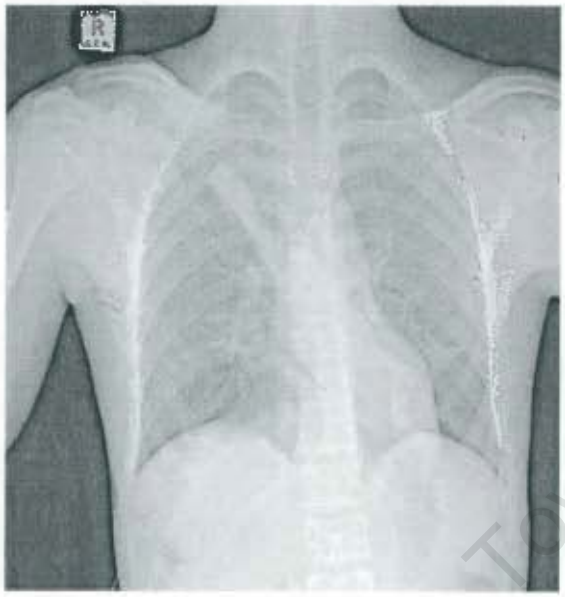

Figure 3.10: Figure 3.2 after being filtered using a Fourier domain high pass filter.

3.5 Wavelet image enhancement

Wavelet analysis has the advantage over Fourier domain analysis in that it copes with signals that change over time. Fourier analysis assumes that the signal remains the same through time and has no beginning or end [27]. This means that any adjustments to one frequency component, in the Fourier domain, will affect the entire signal, often resulting in undesirable distortions (seen in Figure C.10). Wavelet analysis is used to adjust frequencies over a limited range [25, 35, 38, 37].

It is necessary to choose a wavelet for the process, or to create one. The wavelet that is used affects how much the edges are enhanced and the amount and extent of the distortion near the edges. For this project the Matlab tool box on wavelets was used which has a large number of wavelets to choose from. Figure 3.13 show the wavelet referred to as 'sym4' in the Matlab tool box.

Wavelet analysis is a complex field, with many intricacies that are outside the scope of this project. It is however necessary to understand how the wavelets high

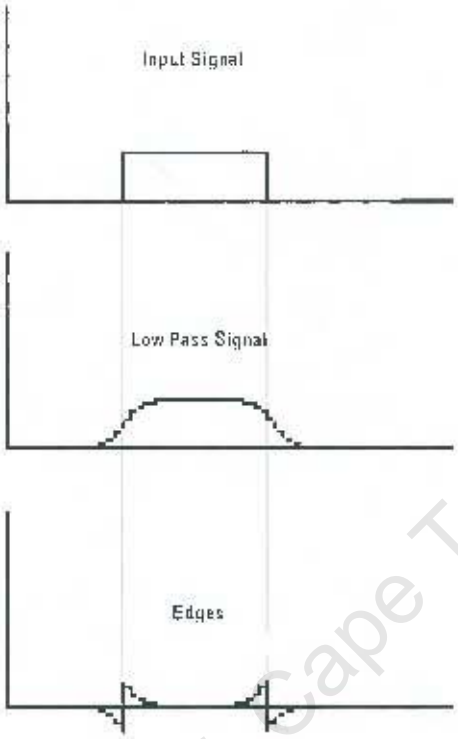

Figure 3.11: Finding edges using unsharp masking

spatial frequencies can be enhanced using this method. Figure 3.14 is a simple signal before it was processed using the wavelet 'sym4' (seen in Figure 3.13). Figure 3.15 shows the low frequency components and Figure 3.16 shows the high frequency components. Notice that both these components are roughly half the width of the original signal, also the high frequency components show the location of edges in the original. Thus by multiplying the low frequency components by a number between 1 and 0 it was possible to enhance the edges in the original, as seen in Figure 3.17. Notice that although the edges were distorted, the contrast between the pixels at the edges were considerably higher than in the original signal. This is in fact, the purpose of edge enhancement.

When a discrete wavelet transform (dwt) is performed on an image, such as Figure

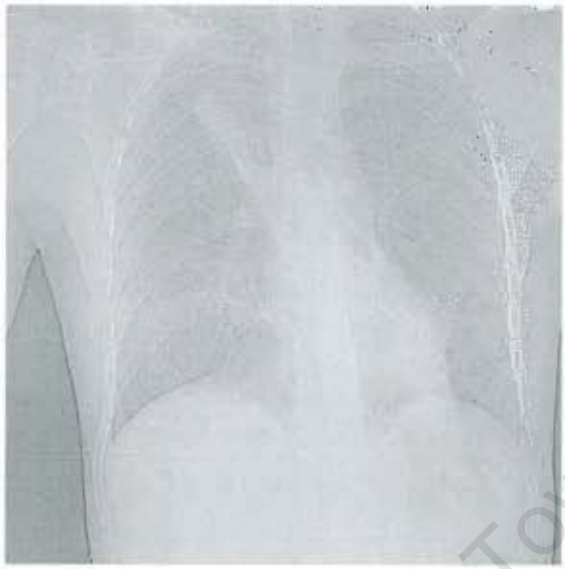

Figure 3.12: Unsharp masking used to enhance detail from Figure 3.2

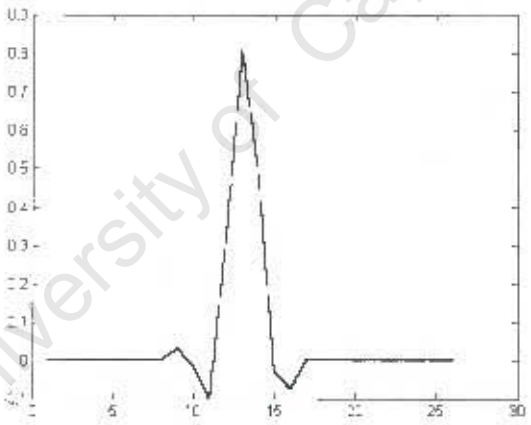

Figure 3.13: An example of a wavelet.

3.18, four new images are created. Each new image a quarter of the area of the original image. Each new image represents a specific set of wavelet components. One represents the low frequency components, and the other three represent the various high frequency components (horizontal components, vertical components

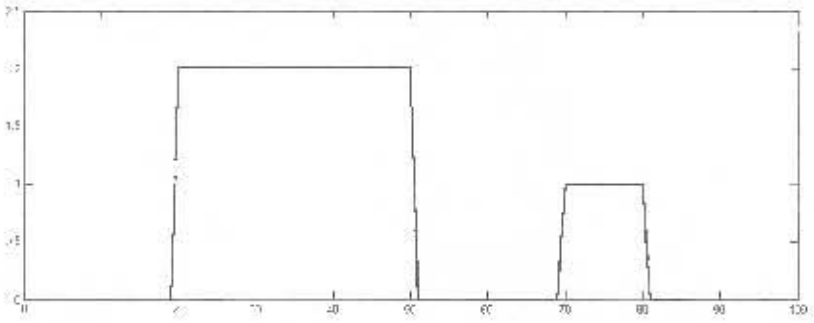

Figure 3.14: A simple signal.

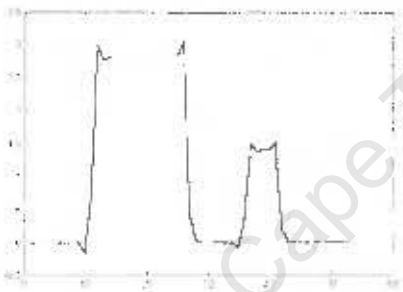

Figure 3.15: The low frequency components from Figure 3.14.

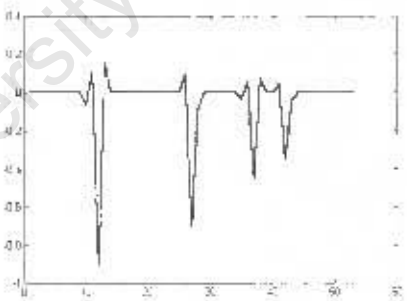

Figure 3.16: The high frequency components from Figure 3.14.

and other high frequency components not covered by the first two). Figure 3.19 shows the low frequency image in the bottom left, and the three high frequency images in the top left, top right and bottom right.

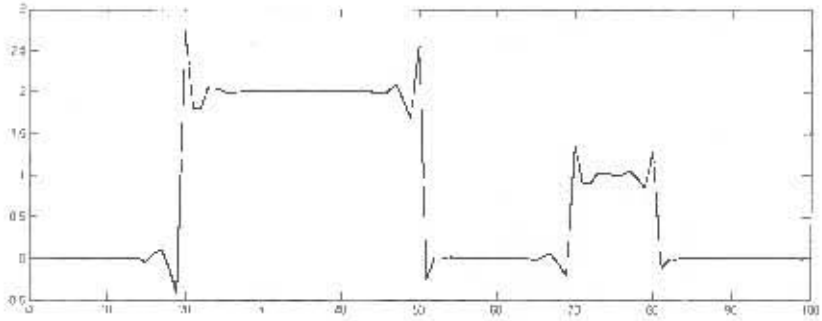

Figure 3.17: The reconstructed signal after suppressing the low frequency components shown in Figure 3.15.

It is possible to create the dwt components for any or all of these component images, and then to repeat this process until the resulting images consist of four pixels each. This process is reversible, so it is possible to perform an inverse discrete wavelet transform (idwt) which recreates the original image from all of these components.

As the high frequency components clearly represents the edges in the image it is easy to enhance the edges in the original image by multiplying the low frequency components by a number between 1 and 0 and then to use the idwt to view the results.

Figure 3.20 shows how an image is broken down into its wavelet components. LL refers to the low frequency components, HL refers to the horizontal high frequency components, LH refers to the vertical high frequency components and HH has the high frequency components that are not covered by HL and LH. It is now possible to compute the level two components by choosing one (or more) level one component images and performing the dwt on them. Figure 3.20 shows how only the low frequency components could be further processed. LLLL refers to the level two low frequency components of LL, LLHL refers to the level two horizontal high frequency components of LL, and so on.

Using the wavelet 'sym4' from the Matlab toolbox, Figure 3.21 was broken down into its different wavelet components as far as level four of the low frequency

Figure 3.18: A picture of a taxi

Figure 3.19: The four wavelet component images created from Figure 3.18

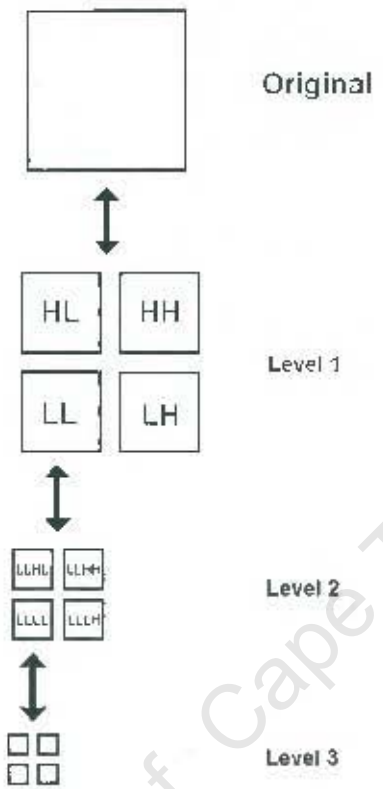

Figure 3.20: The further wavelet component levels into which an image can be broken down.

components. Each low frequency component image was then multiplied by a number between 1 and 0. LLLLLLLL was multiplied by 0.75 then the idwt was performed to recreate the LLLLLL components. These were then multiplied by 0.6 to create LLLL, which was multiplied by 0.5. Eventually LL was multiplied by 0.4. The result can be seen in Figure 3.22. These values, and the wavelet that was used, were chosen by filtering an image and adjusting the values until the desired results were achieved. The criteria for judging the results were the same as in section 3.3.

Figure 3.23 was created using the wavelet 'sym16', in conjunction with the same sequence of low frequency suppression constants. The resulting vein texture was

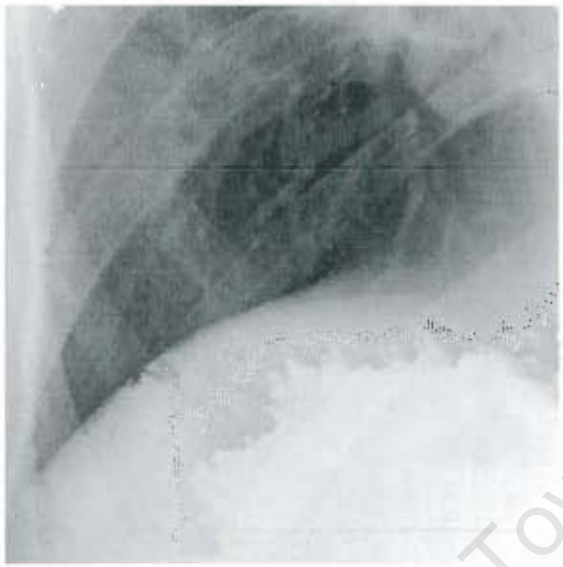

Figure 3.21: An unprocessed section of lung.

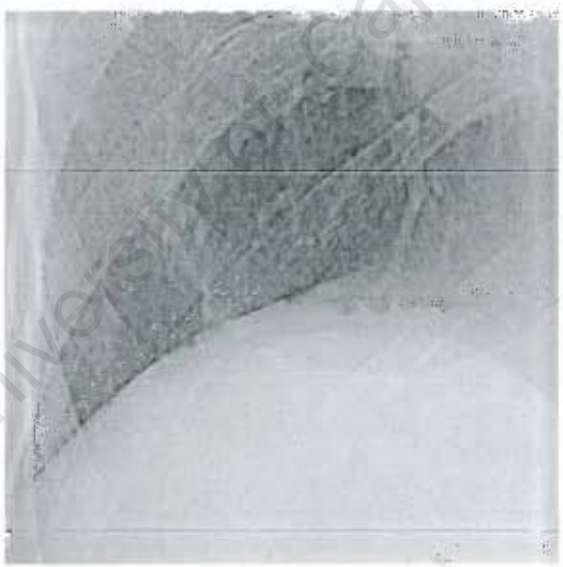

Figure 3.22: Figure 3.21 after being processed using the wavelet 'sym4'.

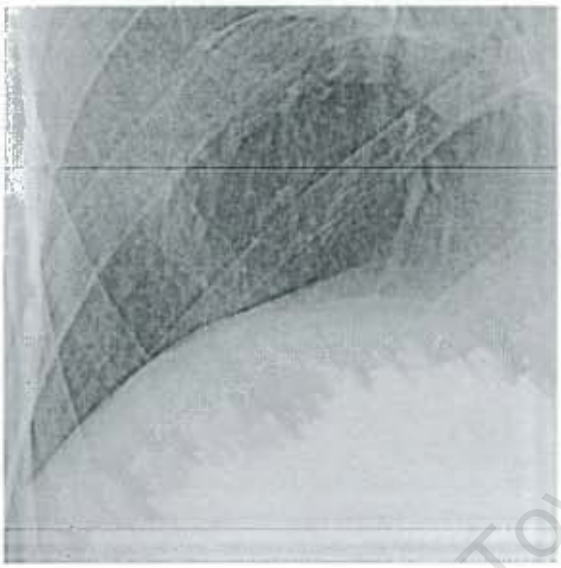

Figure 3.23: Figure 3.21 after being processed using the wavelet 'sym16'.

enhanced slightly better than in Figure 3.22, but the edge distortion, seen flanking the borders round the outside of the image, was considerably worse.

3.6 Rolling Ball Algorithm

One of the problems associated with examining the lung textures, is that they are often obscured by other anatomy. An obvious example of this are the ribs. The rolling ball algorithm [21] is designed to remove objects greater than a certain size, while leaving all other objects relatively undistorted.

Figure 3.24 demonstrates the rolling ball algorithm on a signal that varies in one dimension. A second curve (grey line) is created by fitting a ball under the input curve and tracing the upper surface of the ball as it moves along the signal. The size of the ball stops it fitting into the smaller structures of the signal, so that when the new curve is subtracted from the original signal, all that remains are these smaller structures. The resulting curve can be seen at the bottom of Figure 3.24.

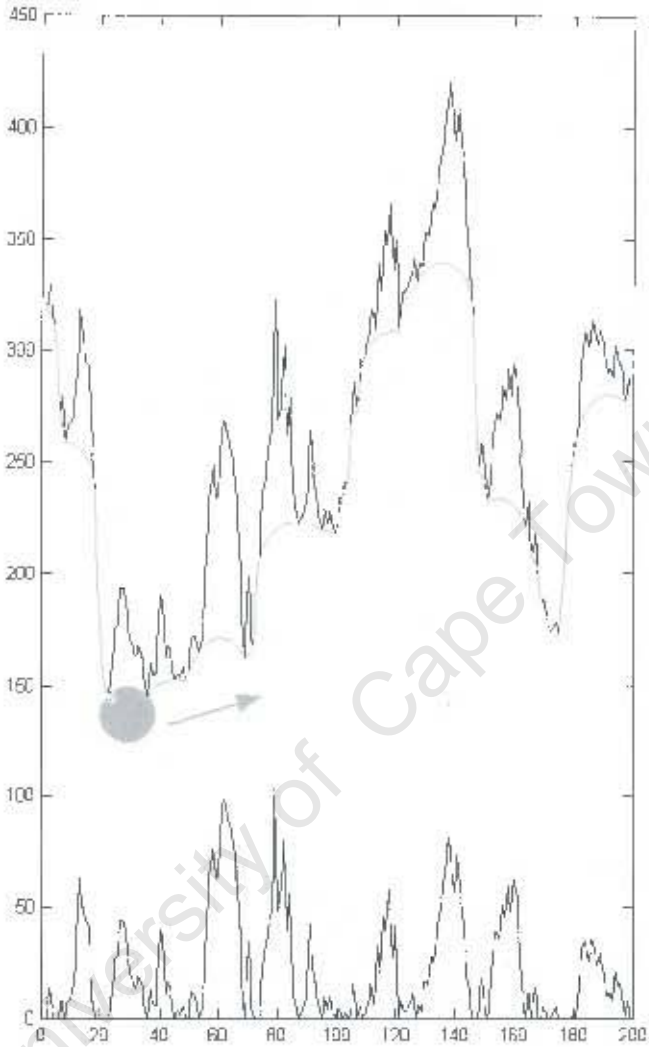

Figure 3.24: A signal before and after being processed with the rolling ball algorithm. The algorithm fits a ball element under the signal, creating the grey line. When the grey line is subtracted from the signal, the bottom signal is created. This removes all structures from the signal that are larger than the ball.

The pixel values in an image can be considered, in a similar way, to be a surface. The ribs in the lung images are larger in structure than the texture of the lungs. By

choosing a suitably sized ball, with a diameter just smaller than the width of the ribs, it was possible to almost completely remove the effect of the ribs from the image.

Figure 3.25: A section of lung from a person with miliary TB

Figure 3.26: A section of lung from a healthy person

A considerable problem with this algorithm is it is very computationally intensive and therefore takes a considerable amount of time to process even small sections

Figure 3.27: The surface created by using a rolling ball algorithm on Figure 3.25. This image was subtracted from the original (Figure 3.25) to create Figure 3.29

Figure 3.28: The surface created by using a rolling ball algorithm on Figure 3.26. Subtraction of this image from Figure 3.26 produced Figure 3.30.

of texture. As a result, this method was not investigated further, as it would take several hours to process a full chest X-ray on a modern computer.

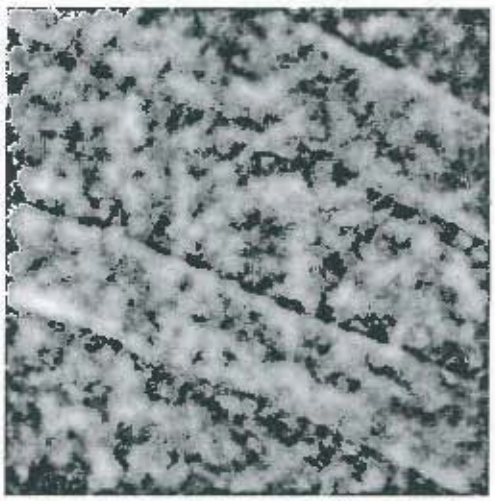

Figure 3.29: The image resulting from subtracting Figure 3.27 from Figure 3.25. The ribs have been almost completely removed from this image.

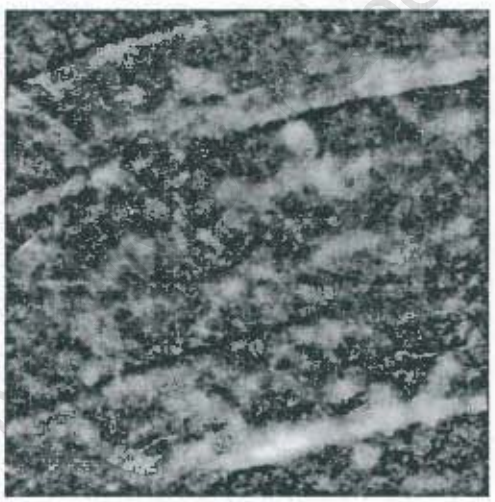

Figure 3.30: The image resulting from subtracting Figure 3.28 from Figure 3.26. As in Figure 3.29. The ribs in this image have been almost completely been removed from this image as well.

3.7 Conclusions of image enhancement techniques.

The types of image processing techniques described in this chapter were adaptations of well known techniques aimed primarily at aiding the human visual system

(HVS) to perceive details that were either beyond the range of the HVS's capabilities, or were obscured in the original signal by high intensity large scale (ie. low frequency) data.

These techniques exaggerated the physiological processes used by the eye itself to extract only the important data from the visual image. In order for them to do this, they had to distort the images. These processes should however not distort detail that was of interest to the observer. Thus there was always a trade off between enhancing one feature and distorting another.

Of all the techniques described here, wavelet enhancement seemed to produce the most promising results. The rolling ball algorithm also showed promise, but was too slow to be used on a large matrix such as a full chest X-ray image. None of the techniques, however, attempt any form of diagnosis-making. The chapters that follow describe diagnostic tasks that would normally be performed by a radiologist.

Chapter 4

Statistical texture measuring techniques

Image enhancement can enhance details in an image to help the radiologist. However even with the very best tools available, people can still make mistakes. Providing a radiologists with a second opinion, that gives consistent results, and does not suffer from fatigue or boredom, would greatly reduce the number of mistakes made [7].

This chapter covers two statistical methods that measured different properties of lung textures. These properties were beyond the capability of the human eye to measure. Howèver computers are ideally suited to measuring these properties.

4.1 Granulometry

Luc Vincent has used granulometry for measuring textures [24]. He used this method successfully in the computer aided diagnosis of silicosis. Silicosis is a lung disease that coal miners get. As silicosis also produces diffuse lung shadowing, the problems of detection are similar to those of miliary TB, which was the focus of this project. This method was used on the lung images.

Granulometry measures the number of pixels a certain sized line segment adds to a texture. When the contributions of all line segments are measured, these values are plotted as a granulometry curve. If two textures differ enough in structure, the resulting granulometry curves should differ significantly.

The following example shows how the granulometry curves are calculated. Figure 4.1 shows a line of pixels from an arbitrary texture. The structure of the texture can be thought of as being made up of a number of line segments lying on top of each other. The principle of this method is to measure the number of pixels each line segment length adds to the texture. It is also possible to measure the line segments needed to fill the gaps in the texture as seen in Figure 4.1. These values are plotted on the negative sides of the graph (Figure 4.2). As a brief example, in Figure 4.1, the line segment of length 3 fits into the graph three times, thus adding nine pixels to the texture so the value 9 is plotted at the location 3 on the graph in Figure 4.3. In Figure 4.2 the line segment of length 5 occurs only once to fill a gap in the texture; so the value 5 is plotted on the graph (Figure 4.3) at the location -5. This process can be used on both columns and rows of pixels. As the textures in the lung are not direction specific, curves for both columns and rows were calculated and then added together.

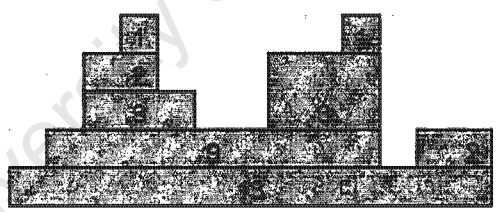

Figure 4.1: An example of how pixel values can be divided into line segments

Luc Vincent used this technique to diagnose silicosis in lung textures. He obtained substantially different curves from the affected lungs than he did from healthy lungs. Computer aided diagnose in this case was very easy. However when the miliary TB textures and the healthy lung textures, used in this project, were processed in this manner, they produced curves that could not be used to distinguish between healthy and unhealthy images. Figure 4.4 and 4.6 show the curves pro-

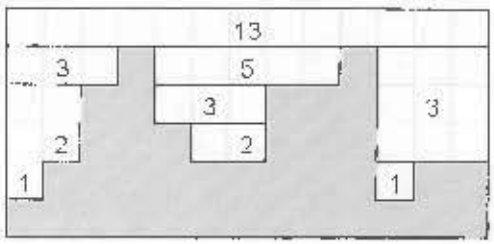

Figure 4.2: An example of how gaps between pixel values can be divided into line segments

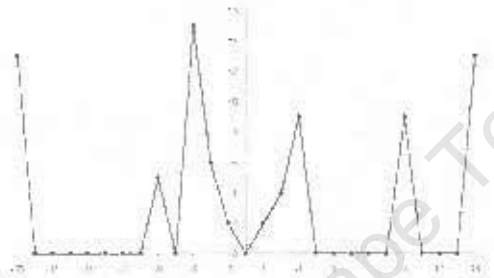

Figure 4.3: A plot of the line segment contributions to the texture in Figures 4.1 and 4.2.

duced by two different sections of miliary TB from the same lung. Figure 4.5 and 4.7 show the curves produced from two different sections of a healthy lung. The curves were only plotted as far as the line segment lengths of 40, as the width of the miliary TB lesions never exceeded 20 pixels. The curves from the same lung differed significantly from each other and there did not appear to be any obvious features in either the healthy curves or the unhealthy curves that would allow the computer to distinguish between them. The curves for the rest of the data set varied even more than the data of these four examples.

The miliary texture is superimposed on top of all the other lung texture and each lesion is superimposed on top of other lesions. This means that when we use the granulometry method to measure the texture, the algorithm measures information other than that necessarily associated with the texture associated with the TB infection. In an attempt to remove some of this extraneous information we

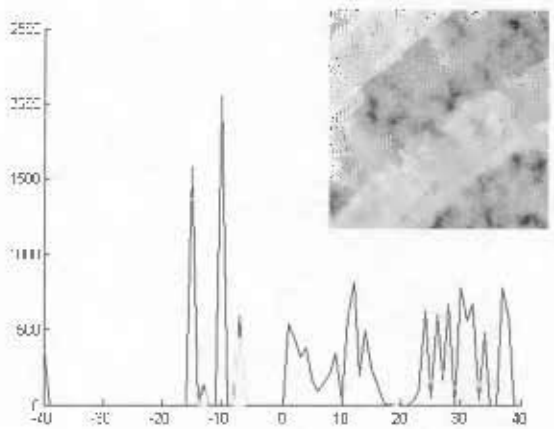

Figure 4.4: The granulometry curve for a section of lung with miliary TB.

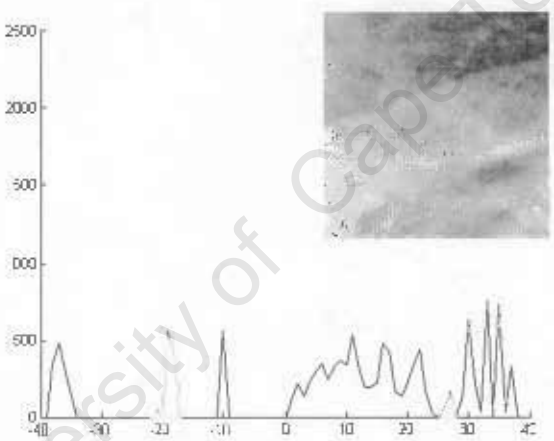

Figure 4.5: The granulometry curve for a healthy section of lung.

used an unsharp-masking filter to enhance edges and then reprocessed the images however this also failed to extract the desired information. Another problem associated with the images used in this project is that although the lesions of miliary TB are all of a similar size in each image, they do vary from image to image. So, if we could measure the lengths of pixel rows making up the texture for one image, it would not necessarily follow that another image would have a similar granulometry curve.

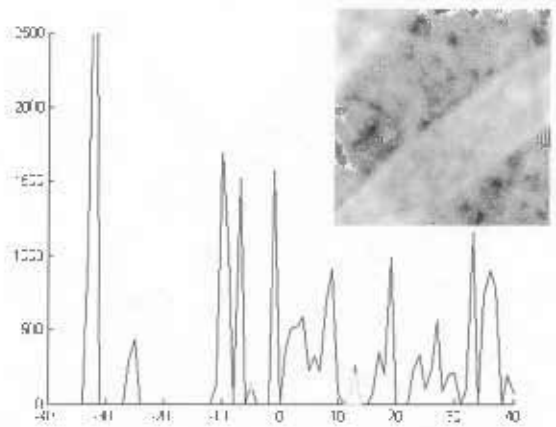

Figure 4.6: The granulometry curve for a section of lung with miliary TB.

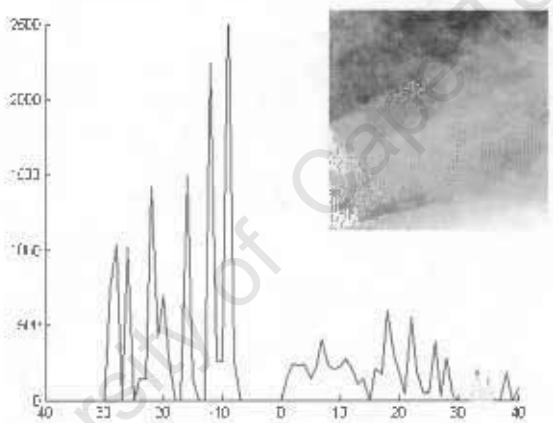

Figure 4.7: The granulometry curve for a healthy section of lung.

4.2 Co-occurrence matrices

Co-occurrence methods measure the proximity of pixels to each other [36]. The resulting measure of a texture is always in the form of a square matrix. The values in the matrix represent the statistical probability that two pixel values will occur at a certain proximity to each other. This method has had significant results in classifying textures in aerial photography. The co-occurrence matrices for lung

textures will be calculated and evaluated to see if a statistical difference can be measured.

In order to understand how the Co-occurrence matrix is calculated, a simple texture will be evaluated using this method. Figure 4.8 illustrates a texture in four grey levels. A number represents each level, white is represented by the value 4, black by the value 1, and the in-between grey levels as either 2 or 3 (Figure 4.9).

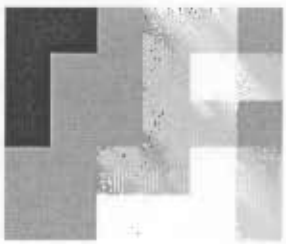

Figure 4.8: A texture with a low spatial frequency.

1	1	2	3	3	2
1	2	2	3	4	3
1	2	2	3	3	2
2	2	3	3	4	3
2	2	4	4	4	3

Figure 4.9: An example of how numbers can represent the grey levels in Figure 4.8.

The co-occurrence matrix for this texture is represented by a 4x4 matrix. This is because the largest value in the texture is 4. The co-ordinates in the co-occurrence matrix represent the values in Figure 4.9 that are being compared. For example, the result of counting the number of times 2's and 3's are next to each other, would be placed at the location (2, 3). As this is the same as counting the number of times 3's are next to 2's the same value would be placed at the location (3, 2).

We can look at values that are horizontally, vertically or diagonally next to each other.

As an example we will create the horizontal co-occurrence matrix for the texture above. We start by looking at the number of 1's horizontally next to each other.

In the top left corner of the texture there is a pair of 1's next to each other. The left 1 is next to the right 1, but the right 1 is also next to the left 1, so therefore this pair must be counted twice and the value 2 is put in the co-occurrence matrix at the location (1, 1).

There are three sets of 1's horizontally adjacent to 2's, so the value 3 must appear in the matrix at the location (1, 2). It is also true to say that there are three 2's next to 1's, so the value 3 must appear in the location (2, 1) in the matrix. There are no 1's next to 3's or 4's so the locations (1, 3), (3, 1), (1, 4), (4, 1) all have the value 0. All possible combinations of 1's and their neighbors have been counted. Next 2's and their neighbors will be counted.

There are four pairs of 2's next to one another. So, as before, with the 1's, these are counted twice and the value 8 is placed in the location (2, 2). There are six 2's next to 3's and so the locations (2, 3) and (3, 2) each have the value 6. There is only one 2 next to a 4 so the locations (2, 4) and (4, 2) have the value 1.

Moving on to the 3's. As there are three pairs of 3's next to each other, the value 6 is put in the location (3, 3). There are five 3's next to 4's so the locations (3, 4) and (4, 3) each have the value 5.

Finally 4's can be counted. there are three 4's in a row at the bottom of the texture, the first 4 is next to the middle 4 which in turn is next to the left 4 and the right 4, and the right 4 is only next to the middle 4, so we put the value 4 in the location (4, 4).

Figure 4.20 demonstrates the co-occurrence matrix for the texture in Figure 4.19, which has a higher spatial frequency than Figure 4.9.

Notice that most of the information in the co-occurrence matrix of Figure 4.9 was situated along the diagonal from the top left to bottom right, the texture in Figure 4.19 has no information along the diagonal in the co-occurrence matrix (Figure 4.20). The first texture (Figure 4.9) has large numbers of pixels with the same value next to one another, while in Figure 4.20 there isn't a single pixel

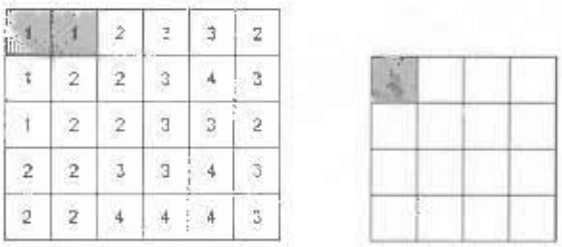

Figure 4.10: The texture and its co-occurrence matrix on the right showing how the matrix is calculated by counting all the 1's next to 1's.

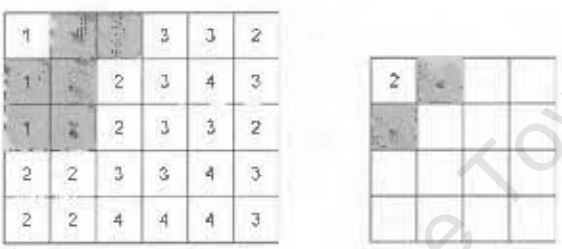

Figure 4.11: The texture and its co-occurrence matrix on the right showing how the matrix is calculated by counting all the 1's next to 2's.

Figure 4.12: Because there are no 1's next to 3's or 4's the matrix has 0's in these locations.

next to another that has the same value. This therefore demonstrates a method of distinguishing different textures from one another.

Other variations on this technique are possible. For example instead of looking at adjacent pixels, pixels that are several pixels apart can be compared. This is referred to as varying the spatial resolution of the matrices. This makes it possible to measure different spatial frequencies in textures. Another method for obtaining

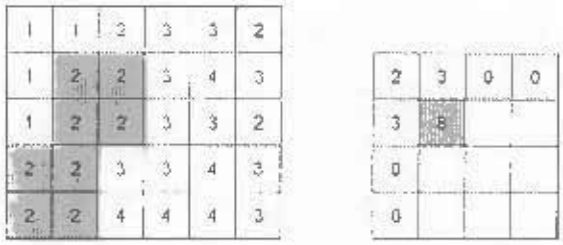

Figure 4.13: The texture and its co-occurrence matrix on the right showing how the matrix is calculated by counting all the 2's next to 2's.

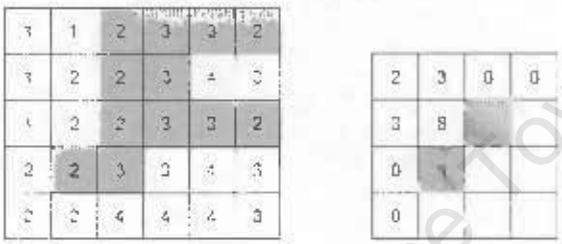

Figure 4.14: The texture and its co-occurrence matrix on the right showing how the matrix is calculated by counting all the 2's next to 3's.

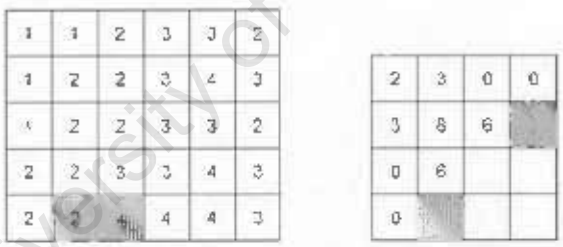

Figure 4.15: The texture and its co-occurrence matrix on the right showing how the matrix is calculated by counting all the 2's next to 4's.

similar spatial information would be to resize the image (i.e. make the image smaller).

Differences between co-occurrence matrices can be very subtle, and a statistical approach is the usual method for evaluating them [36]. It was decided to evaluate the co-occurrence matrices by means of three statistical measures as it is easy to

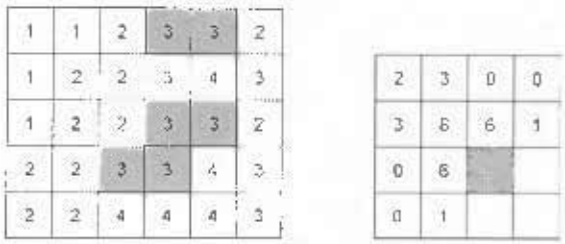

Figure 4.16: The texture and its co-occurrence matrix on the right showing how the matrix is calculated by counting all the 3's next to 3's.

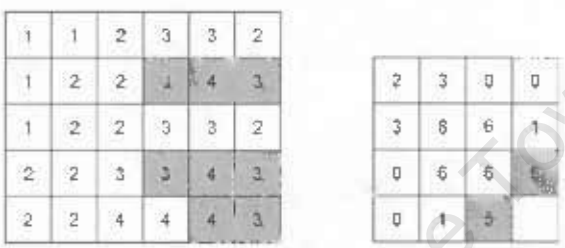

Figure 4.17: The texture and its co-occurrence matrix on the right showing how the matrix is calculated by counting all the 3's next to 4's.

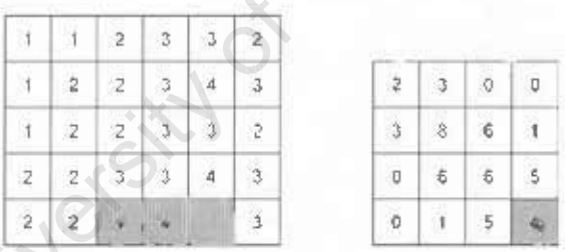

Figure 4.18: The texture and its co-occurrence matrix on the right showing how the matrix is calculated by counting all the 4's next to 4's.

plot three measures in relation to each other. The Standard Deviation (SD), entropy and maximum probability of each of the resulting matrices were calculated.

The maximum probability is simply the maximum value in the matrix as each value in the matrix represents a probability. The quantities were chosen based on their simplicity, their ease to implement and their independence from each other.

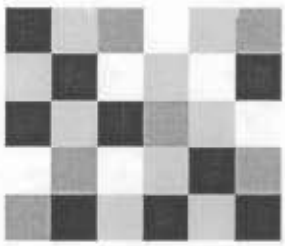

Figure 4.19: A four colour texture, with a high spatial frequency.

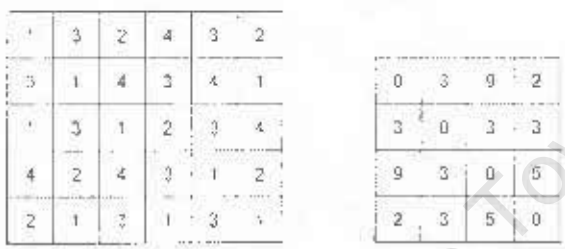

Figure 4.20: An example of how numbers can represent the colours in Figure 4.19, and its co-occurrence matrix.

If these measures show any significant results more can be evaluated.

Entropy	$\sum x * \log(x)$	x = each value in a matrix

Small 100x100 pixel sections of lung texture were chosen from a number of images so that there were 28 images in the healthy set and 28 images in the miliary TB set. The bigger the lung sections used to create the matrix, the more general the matrix will be, but the longer it would take to create. Pixel blocks of 100x100 pixels were therefore chosen as a compromise. Forty different matrices were created for each lung section by varying spatial resolution from 1 to 40. Each of these matrices were then evaluated using the statistical measures described above.

Figure 4.21 shows the measures of entropy, standard deviation and maximum probability plotted against each other, for the matrices calculated using a spatial resolution of 1.

The three statistical tests clearly evaluated the data in similar ways as each set of

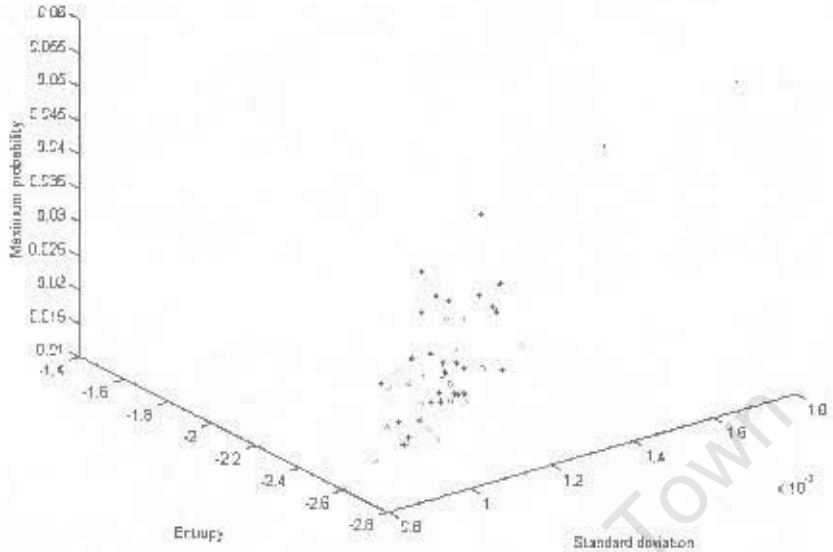

Figure 4.21: A plot of the healthy values vs the TB values. The TB values are black crosses and the healthy values are grey circles.

results had a high correlation with the other two. The correlation coefficient for the Entropy and maximum probability was 0.802, for maximum probability and SD it was 0.707, and for entropy and SD it was 0.828. Furthermore, it was also clear that the healthy data and unhealthy data could not be separated in this plot. Each measure was then separately plotted as a ROC curve. Figure 4.22 shows the entropy ROC curve, Figure 4.23 shows the standard deviation curve and Figure 4.24 shows the ROC curve for the maximum probability. None of these curves vary materially from the diagonal line, which represents the results that would be generated by a purely random data set. Varying the spatial resolution from 1 to 40 did not produce significantly different results. This method was therefore judged to be unsuccessful for distinguishing between healthy an unhealthy data sets.

The lung textures in both the healthy and unhealthy data sets were made up of similar combinations of pixel values. This method relies on textures that are dependent on the proximity of pixel pairs. The textures in these data sets are apparently not dependent on pixel pairs. As a result another method that measures

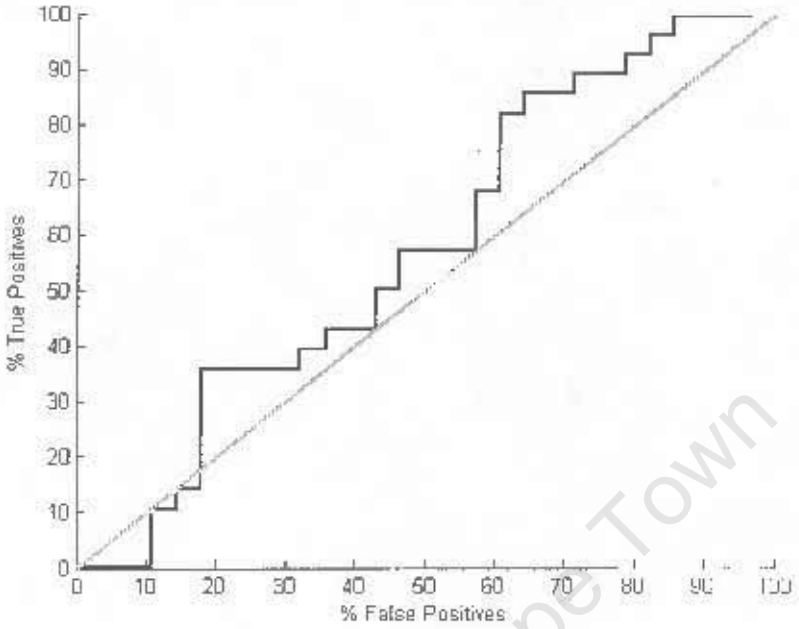

Figure 4.22: The ROC curve for the entropy measures from the data set.

larger groups of pixels has been evaluated. This is be described in the next chapter.

4.3 Conclusions for Statistical texture measuring techniques

Statistical methods can be used to measure features that a human eye is incapable of measuring accurately. The two methods discussed here failed to produce significant results. It is considered that the reason for these methods failing is that they do not measure the shape of the miliary TB lesions, and that the features that they do measure are common to both healthy and unhealthy textures.

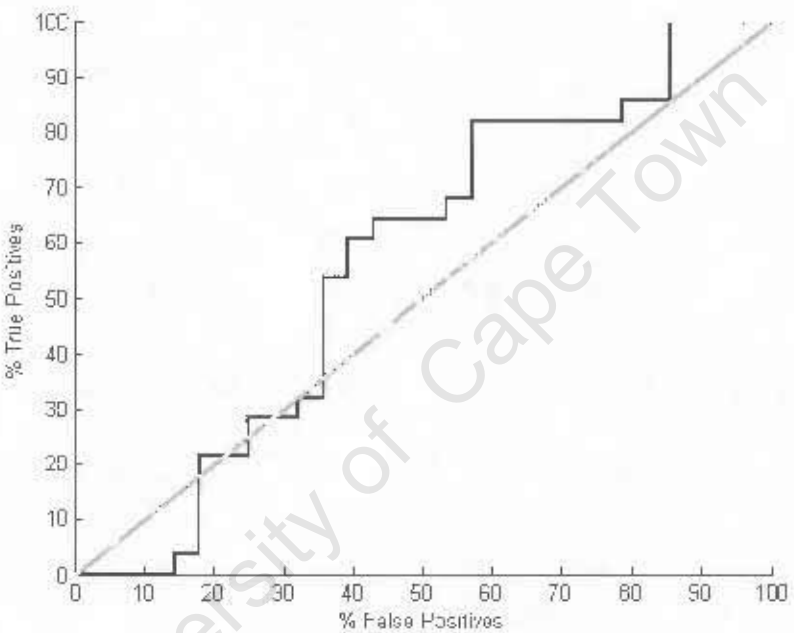

Figure 4.23: The ROC curve for the standard deviation measures from the data set.

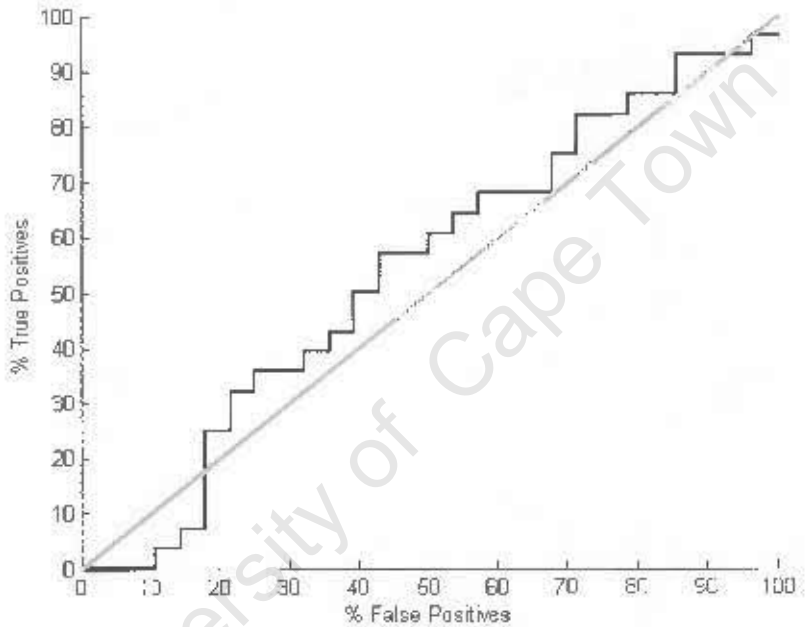

Figure 4.24: The ROC curve for the maximum probability measures from the data set.

Chapter 5

Template Matching

The statistical methods, discussed in the previous chapter, proved to be unsuccessful at distinguishing miliary TB from normal lung. Since the eye can readily distinguish between them, it is probable that this is not the method used by the brain to perceive the textures. Template matching is an attempt at doing something that is probably more closely related to what the human visual system (HVS) does to detect the presence of miliary TB.

The pulmonary manifestation of miliary TB lesions looks like a number of light and dark translucent spots superimposed on top of the the lung texture in a chest X-ray. Figures 5.1 and 5.2 are examples of a healthy lung and a lung with miliary TB, respectively. If the textural shape of one of the miliary spots can be estimated, it is then possible to search through the rest of the image looking for all the other spots that have the same or very similar shape. This process is called template matching. The limitation is that the template must have the same size and be in the same orientation as the object it is being matched with or the computer will not confirm the match, even if it is obvious to the human eye.

Figure 5.1: An example of an X-ray image of a healthy section of lung.

5.1 Estimating the Template

The first step is to estimate a template from the texture. This was done by looking at the pixels that make up one of the lesions. Figure 5.3 is an enlargement of a section of lung with miliary TB. It is enlarged enough that each pixel can easily be distinguished from its neighbor. The left of the two circles, marks a bright group of pixels that represents a miliary TB lesion. This particular spot was chosen because it is slightly isolated from other similar spots; in other words it does not overlap any other light spots. The shape of this lesion looks roughly circular with a diameter of approximately 8 pixels. The brightness of the pixels in the lesion range from darkest near the circumference to brightest in the centre. Selecting a

Figure 5.2: An example of a section of lung that has miliary TB.

pixel row from the image gave clear a indication of the shape of these lesions. These lesions have an approximately standard size per image although they do vary in size from image to image. In the original plates the sizes varied from 3mm to 7mm. The lesions are evenly spread throughout the lungs. This means however that in the X-ray image, where the X-rays pass through more lung tissue, the lesions will appear to be more densely concentrated. The circle on the right, in Figure 5.3, shows an area of dark pixels. Areas like this represent the spaces between the miliary lesions, and are a feature of the texture as well. These areas, surprisingly, but very consistently, tend to have roughly the same diameter and shape as the light lesions.

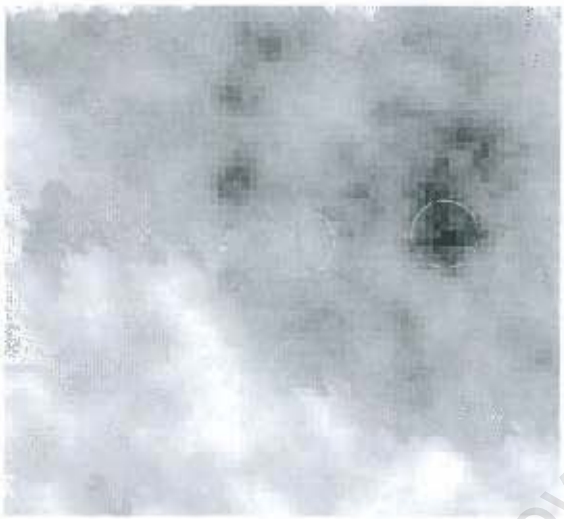

Figure 5.3: A close up view of the miliary TB lesions.

Each pixel has a value that represents the brightness of that pixel. Bright pixels have larger values than darker pixels. Plotting these values for a single row of pixels gave an indication of how the brightness varies across a portion of lung. Figure 5.5 is a plot of a row of pixel values from a lung with miliary TB. The lesions show up as peaks. As there are many veins and air ways in the lungs the shape of these lesions are distorted and so the shape of these peaks was generalized. A parabolic peak, shown in the plot, was estimated to fit into the peaks in the plot. The same parabolic shape, when inverted, was found to fit into the valleys between peaks. For comparison Figure 5.4 shows a plot of a row of pixel values in a normal lung. Figure 5.6 shows the estimated template of a miliary lesion. Here the amplitude of the template was set to 1. The variation in amplitude of the miliary lesions on a given X-ray plate (Figure 5.5) was found to be between approximately 100 and 250.

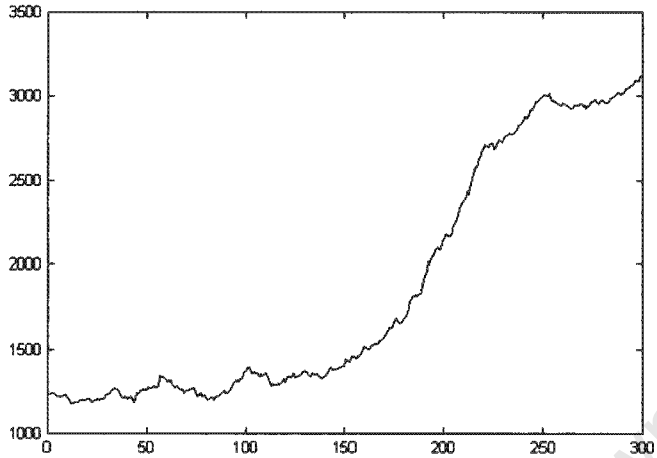

Figure 5.4: An example of a row of pixels from a healthy lung.

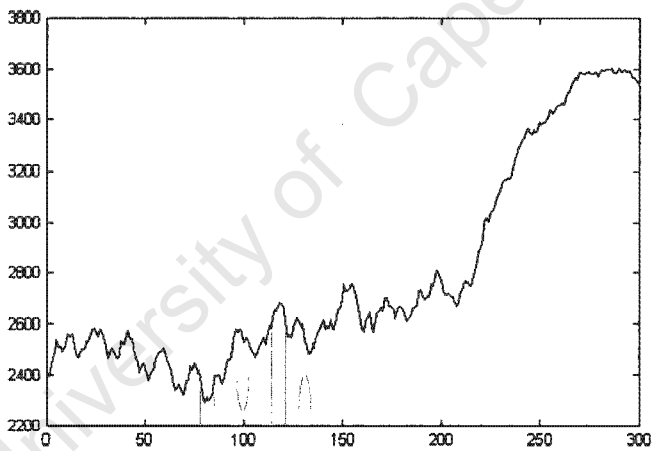

Figure 5.5: An example of a row of pixels from a lung that has miliary TB.

5.2 Simulating TB with a Template

It is important, once the template has been estimated, that the validity of the template is established. This was done by using the template to simulate the texture of miliary TB.

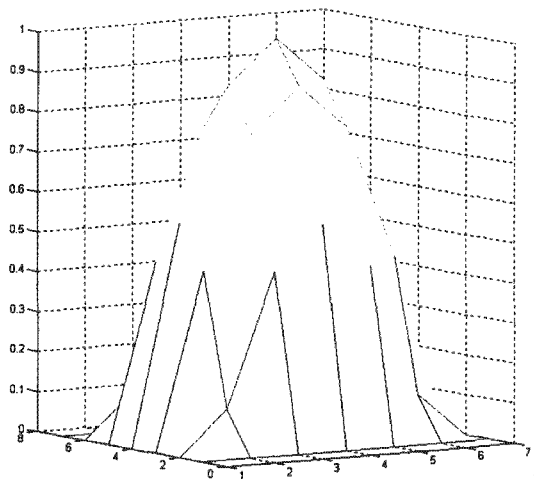

Figure 5.6: The estimated template.

A 300 by 300 pixel section of lung was chosen from an X-ray of a healthy person, The template was then added to this image in approximately 1000 random locations. The amplitude of the template being added was also varied slightly between 250 and 100. This was done for a number of other lung sections from other plates, which were then mixed up with a series of real TB lung sections so that the total number of images was 66. Figure 5.7 is a healthy image with the simulated TB. Figure 5.8 is an example of miliary TB for comparison.

These were set up in a random order, each image with a unique number so that it could be identified later, and taken to Prof. Steve Beningfield, Head of Radiology at Groote Schuur Hospital in Cape Town, for classification. He was informed that some of the images were fake and some were real and he should try to identify the fakes. He spotted several immediately as the template had been added to locations in the image where there was no lung. He also said that the lesions were too bright in some images, although he said this for some of the real TB cases as well. It was therefore assumed that he was not used to looking at the texture of such limited sections of lungs.

Lastly he found that the edges of some of the artificial lesions were too distinct,

and should be blurred slightly. This effect in the real TB images is due to X-ray scatter. The extent of this scatter is a function of the distance of the object from the X-ray detector or film, but is quite small for the lesions of miliary TB.

Figure 5.7: Simulated miliary TB.

Prof. Steve Beningfield's advice, regarding the blurring of the template, was implemented as an averaging filter with a footprint of size *WD/3* rounded down (*WD* is the width of the template). The amount of blurring was chosen to be just enough to soften the edges of the template. It was later found that blurring the template had little affect on the results of template matching. So the type of blurring function was considered to be mostly unimportant. Figure 5.9 shows the template after being blurred.

5.3 Template Subtraction

With the verified template it was now possible to search for objects in the image that closely resemble the template. Template subtraction is a simple and versatile

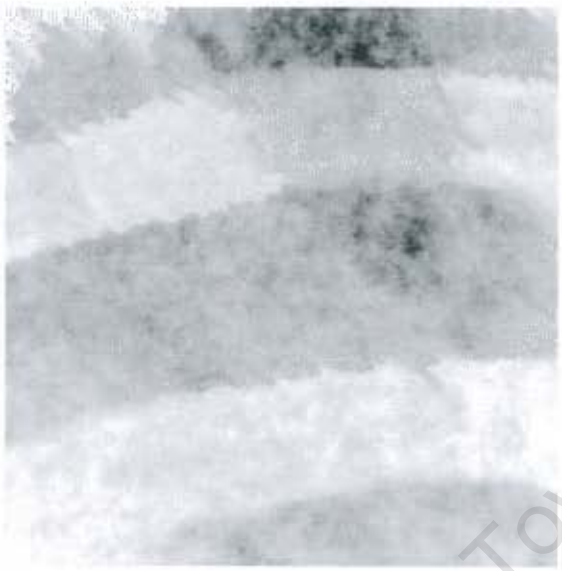

Figure 5.8: Real miliary TB.

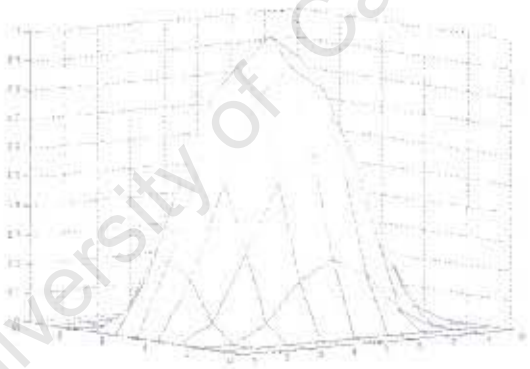

Figure 5.9: The template after being blurred slightly.

method of template matching [21]. Its principle drawback is however that it is very slow.

The subtraction is done as follows. A section of the image with the same dimensions as the template is extracted from the corner of an X-ray plate. first the minimum pixel value is subtracted from all the pixels in this section. This is done

so that when the template as a whole is subtracted from this section there is no error induced by this offset. A new matrix is then formed when the template and the image are subtracted, pixel for pixel, from each other. At this point it is possible to weight the subtraction. Generally weighting is used to add a heavy penalty to each pixel that does not match the template. A common method is thus to square each value in the subtracted matrix. The template was chosen to match an isolated lesion, but most lesions are superimposed on other anatomy or overlap other lesions. It was therefore decided not to weight the values in the matrix, and simply to accept the absolute values of all the subtractions. Adding all the values in the matrix together now gives a value that represents the closeness of the fit of the template to the image at that point. A small value represents a close match and a large value represents a bad match.

This process is then repeated for each possible match in the image. This has to be done by moving the template by one pixel width and then performing the template subtraction as described above. The value from each match is placed in a new matrix at the location of the centre of the section of image that was matched with the template. When the entire plate has been processed, the result is a new matrix with low values for close matches, and large values for bad matches.

A section of lung from an X-ray image of a healthy person and an image of a person with miliary TB, were selected and processed in this manner. The results from one of these images was compared to an arbitrary threshold so that any values below the threshold appear as a white pixels in a binary image. The higher the threshold, the larger the groups of white pixels whenever there is a template match. As a close match should represent a low value surrounded by high values, the threshold was set so that no pixel group was greater in size than 9 pixels in either the healthy or unhealthy image. This was done by using region labelling and then using a histogram to measure the number of pixels in each pixel group. The result can be seen in Figures 5.10 and 5.11 where the results were superimposed on top of the original images. Each dot has been made larger by using a binary dilate. This was done to make the results easier to see in the two images.

As this template has a very general shape, a large number of false positives were present in the resulting images, but despite these false positives, the TB image

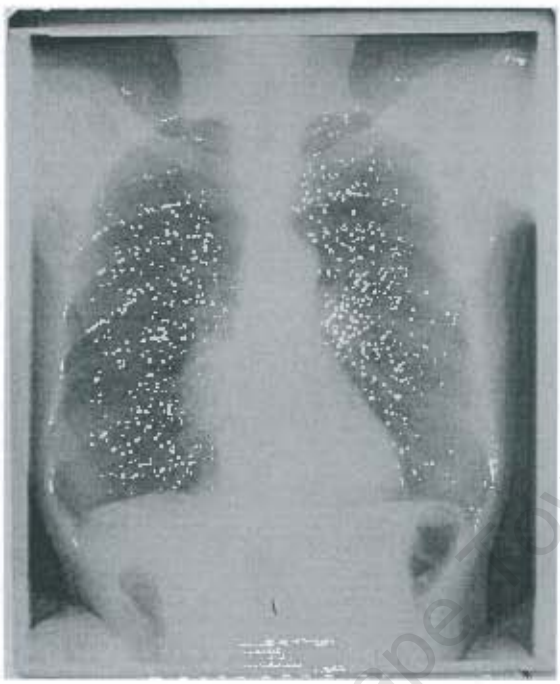

Figure 5.10: Template subtraction performed on a healthy lung

displayed considerably more positives in the lungs than were seen in the healthy lung.

5.4 Fourier Domain Correlation

Although the template subtraction method described above produced favorable results, it is very slow. Fourier domain correlation is considerably faster than the space domain subtraction technique [27, 22]. The method involves converting both the template and the image to the Fourier domain, multiplying the Fourier magnitudes of both images together, and then converting the result back to the space domain.

First, the mean value of the template must be subtracted from all the values in the template, thus the template haves a mean value of zero. If this is not done,

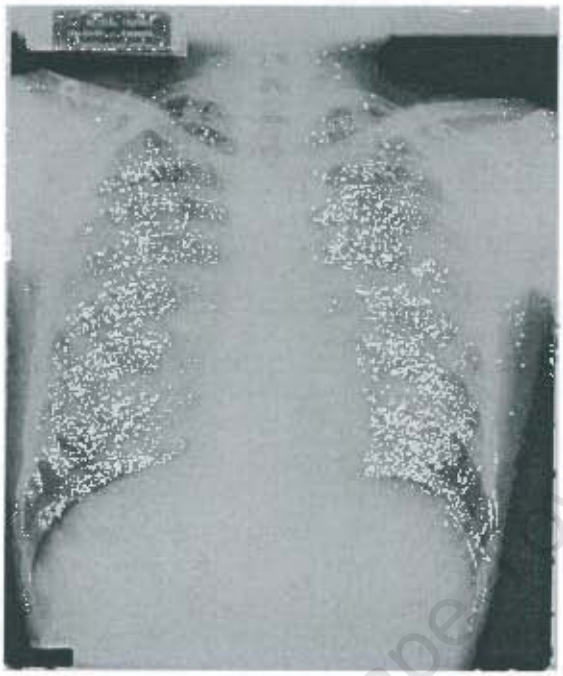

Figure 5.11: Template subtraction performed on a lung with miliary TB

the result will be a blurring and not correlation. The practical difference is that convolution results in a smoothed image with the template as the smoothing function. Correlation requires that the template be padded with zeros. This is done by adding rows and columns of zeros to the template until it has the same dimensions as the section of image it will be correlated with. It is important that the template is normalized before padding, or it will result in a convolution with a very small negative offset. Usually the template must be moved into one of the corners of this padded matrix or the results will be shifted by the distance between the centre of the template and corner of the matrix. In the present study however, this was unnecessary as further processing, described next, meant the shift did not affect the results.

The results from the correlation differ from the template subtraction described earlier. With this method a bad fit will result in a value close to zero, a close fit will result in a large positive number. If the shape of the section of image is the

same as the template, but the amplitude of that section is larger or smaller than the template, then the the resulting correlation will be bigger or smaller accordingly. Another difference is that if there is a section of image with the same shape as the template, but all its values are negative, then the result will have a large negative value. As mentioned earlier, these inverse shapes are also descriptive of the texture of pulmonary miliary TB. After correlating the template with the image section, the absolute value of the result is taken in order to incorporate the inverse template shapes into the results.

A significant problem with these images was that the contrast in the lungs was largely dependent on the developing process, as well as the particular machine used for taking the X-ray images. As the data set was made up of lung segments, each chosen to have only lung texture and the occasional rib, each image had its contrast stretched using a linear histogram equalization before the correlation. The mean of the absolute values from the correlation matrix for each image was calculated and used as a measure of the texture of that image. This process was repeated for each image using a variety of template sizes. Figure 5.12 shows the correlation values for each image as the template size is varied from 4 to 18. The template size refers to the width of the template being smoothed. It should be pointed out that the correlation values were scaled to make them easier to compare. This was done by evaluating the mean value of the template correlated with its self and the correlation results were then divided by this number. This does not effect the separation of the correlation values, but does make them easier to compare to the correlation values of other template sizes. The black lines represent the TB images and the broken grey lines represent the healthy images. From studying the curves in Figure 5.12 it was decided that the template size had little effect on the separation of the correlation values for healthy and unhealthy images. Also it was a very slow process to correlate each image with a variety of template sizes, especially when a full chest X-ray was being processed. In order to decide on a single template size to process all the images the 'student T-test' was used to calculate the probability that the data spread was due to chance for each template size. Although all the template sizes produced results of very low probabilities, template size 9 had the lowest (1.12632e-10) and so it was decided to use this template size to evaluate the miliary TB texture.

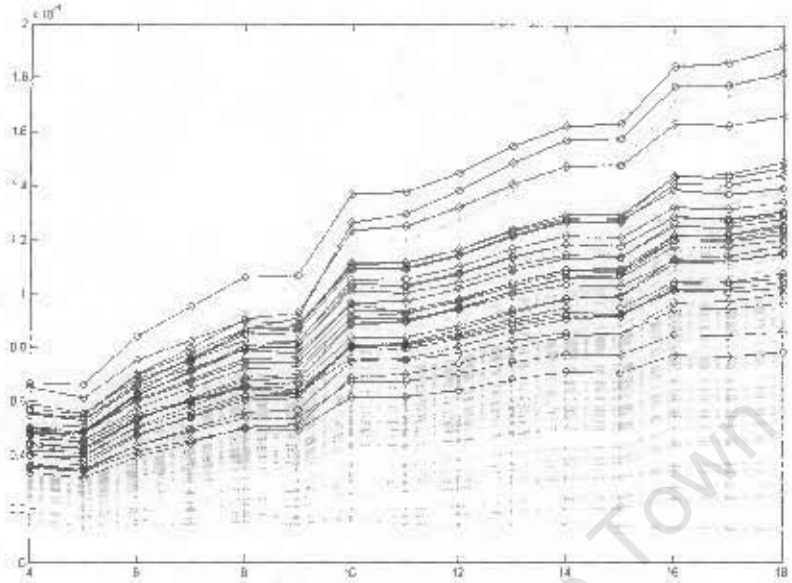

Figure 5.12: Correlation vales for varying template sizes

5.5 Choosing a threshold

In order to distinguish healthy images from the unhealthy images, a threshold had to be decided upon. when a section of lung had a correlation value above this threshold it was marked as having miliary TB. The Receiver Operating Characteristic curve (ROC curve) was used to establish a threshold. Figure 5.13 shows the ROC curve for template size 9, It shows the trade off between false positives and true positives [28]. The grey diagonal line represents a completely random data set and so the further from this line the evaluated data set is the more accurate the classification. The percentage of the area under the curve is used as a rough measure of this accuracy. The area is evaluated as in the following table.

100 – 90%	excellent
90 – 80%	good
80 – 70%	fair
70 – 60%	poor
60 – 50%	fail

The area under the curve in Figure 5.13 is 89.8% and so is considered as being a 'good' classification. The correlation value (6.1015e-005), that minimizes the false positives (14.5%) and maximizes the true positives (87.5%), was chosen as the threshold.

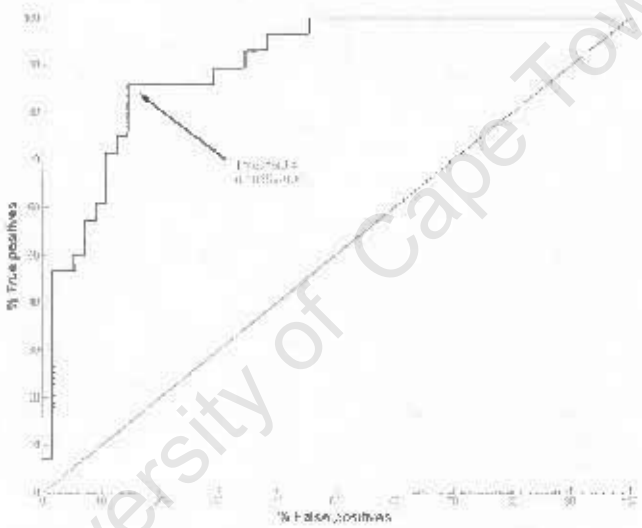

Figure 5.13: ROC curve for template size 9

Figure 5.14 and 5.15 show the results when plotted on full sized chest X-rays. Notice the false positives in areas of the image outside of the lungs in both images. The template shape was a very general shape, and so it picked up features that were similar in shape to itself, but not actually TB lesions.

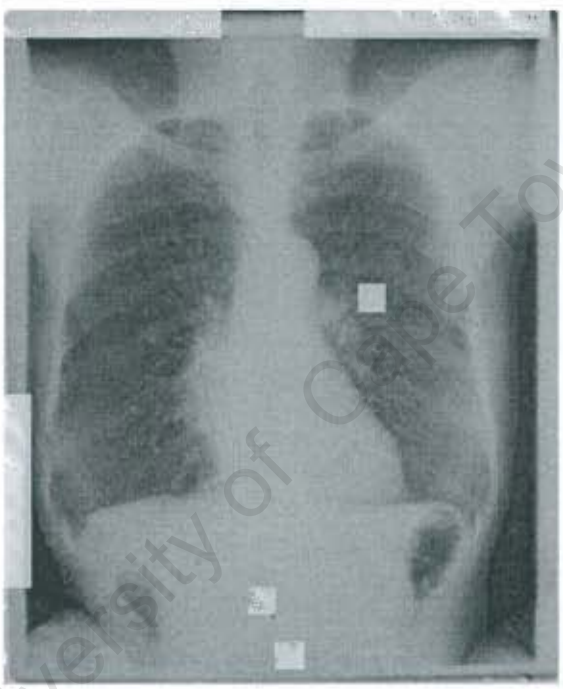

Figure 5.14: Fourier correlation performed on a healthy lung

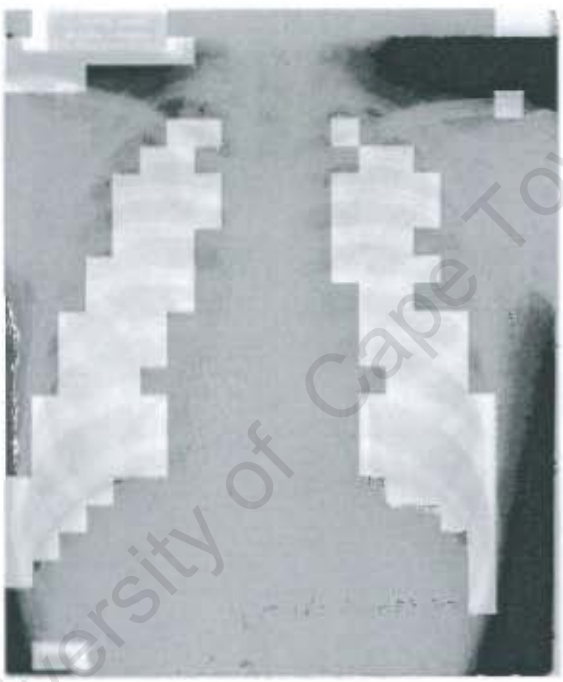

Figure 5.15: Fourier correlation performed on a Lung with TB

Chapter 6

Lung Segmentation

The template matching method discussed in the previous chapter, measured features both inside and outside the lungs, so often a section of image would be marked positive outside the lungs. These are obviously false positives, and therefore should be removed when possible. An obvious method for removing these false positives is to tell the computer not to plot a positive if it is outside the lungs. However it is considerably easier for a person to distinguish between the lungs and the rest of the image. This chapter will evaluate two simple methods for segmenting the lungs from the rest of the image.

6.1 Adaptive Thresholds

In a chest X-ray the lungs appear darker than the surrounding tissue, this is due to the large number of air spaces in the lungs. A threshold can be applied to the pixels in the image to make a black and white image [6, 19]. In this case the threshold was set so that if the pixels had a value above the threshold they were set to black and if the pixels had a value less than the threshold they were set to white. As these images varied significantly in contrast, a simple threshold set at a standard value got different, and often incorrect, results for each image. It was necessary to make the threshold so that it adapted itself to the image. It was found that by raising

the threshold to the point where there were only two separate regions of white, the shape of the lungs were clearly represented. This was done automatically by using region labelling to count the number of separate white regions in the image until two separate regions were counted. Unfortunately there were other regions that were darker than the lungs, for example the regions outside the body and some sections of the colon. These dark spaces had values well below the threshold necessary to segment the lungs and therefore also appeared white when the threshold was applied to the image. These white regions were attached to the white regions that formed the lungs. Figure 6.1 shows a chest X-ray, and the result of thresholding it to segment the lungs out. Although the shape of the lungs were clearly defined in places, this method was not ideal, as it would have been very difficult to remove the other unwanted white regions.

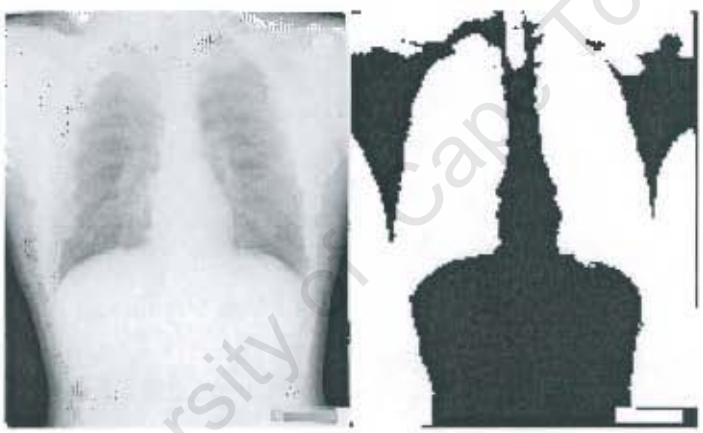

Figure 6.1: A chest X-ray image and the result of using a threshold to segment out the lungs.

Dian [6] used this method on Lodox images to segment out the lungs. Due to the image quality, the lungs were easy to segment and only a little work was necessary to remove unwanted edges. The images from the data set used in this project however varied significantly and the results from this method were not good enough to segment out the lungs satisfactorily.

6.2 The Watershed Function

If the light sections of an image are thought of as being high, and the dark sections are thought of as being low, the image can be thought of as a landscape. The watershed function [20] uses the image in this way and fills the valleys with 'water', creating 'dams'. When two 'dams' start to touch, it builds a 'wall' between them, and continues to fill them. The algorithm stops when the entire image is covered in 'water' and 'walls'. If the starting points at which 'water' is added to the image are selected correctly it is possible to segment a desired region from the rest of the image. The location of the starting points is critical to the success of this method. Both regions of interest and undesired regions must have starting points.

This algorithm was applied to the chest X-ray images in this projects data set in the following manner. First the edge image was created using the method described in section 3.2. This was done so that any starting point placed in a light region of the X-ray would not cause 'water' to leak into one of the lungs.

Figure 6.2 shows a number of starting points for a chest X-ray. A border around the image was drawn as a starting point for an undesired region. Four more starting points were put at locations in the bottom right, top right, top left and bottom left of the image as starting points for undesired regions. These places were found by finding the local minimum of the pixel values in the original image in these locations. Another starting point was plotted in the throat area, this was also done by finding a local minimum in this area.

As the ribs can have significant edges which might cause problems when using the watershed function, a line of starting points were draw down each lung. This was again done by finding local minimums, in the original image, in the region of the lungs. Figure 6.2 shows how this method doesn't always work, as the exact orientation of the patient in the image can be unpredictable. The lines for the region of interest in the lungs have been plotted right down in to the abdomen.

Using these starting points, the watershed function was then run on the edge image of the original image. The regions that were created from the starting points inside the lungs were then used to segment out the lungs. Figure 6.3 shows the result of

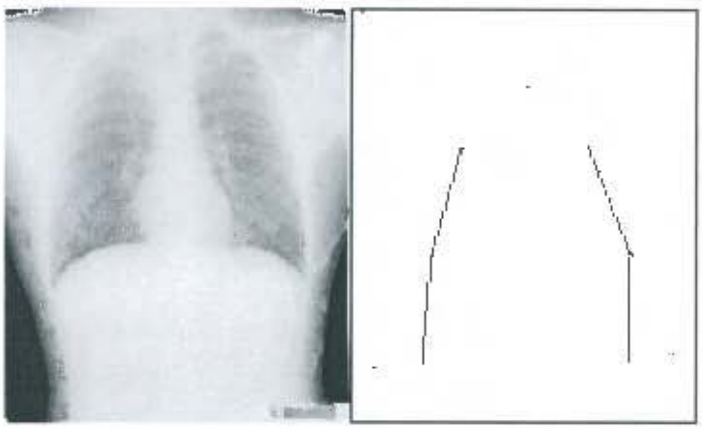

Figure 6.2: A chest X-ray image and the starting points for the watershed function.

segmenting the lungs in this manner. The results show the edges of the lungs well in some regions, but in other regions this method completely failed to find the edges of the lungs.

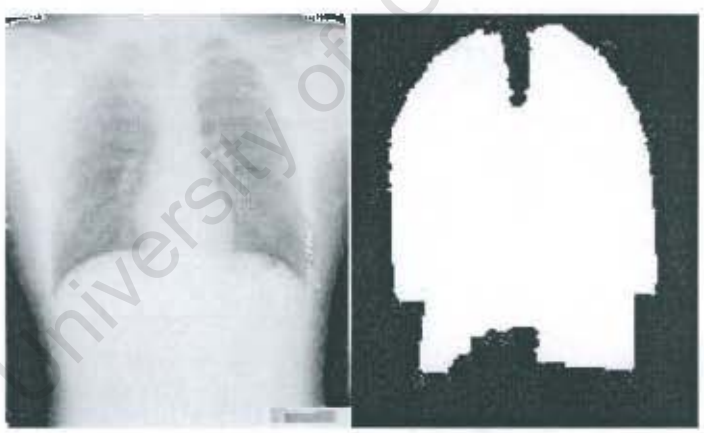

Figure 6.3: A chest X-ray image and the result of using a watershed algorithm to segment out the lungs.

6.3 Combining Thresholding with the Watershed Function

It was obvious to see that the edges of the lungs found using the threshold were different to the edges found using the watershed. As the results from both methods were in the form of binary images, it was easy to find the mutual information in each by performing a binary 'AND'. The effect of combining the results from Figure 6.3 and 6.1 can be seen in Figure 6.4 where the results have been superimposed on the original image. Apart from a small region in the bottom left of the picture, the outline of the lungs is correctly defined.

Figure 6.4: Combining the two methods to segment out the lungs provides a considerably better result. Here the segment lung region has been superimposed on the original image.

Figures 6.5 and 6.6 show the results of segmenting the lungs from two different images. In both figures, the different results of each step are shown. The original image is labelled 'a', the result of thresholding is labelled 'b', the result of the watershed function is labelled 'c', the result of combining the two methods is labelled 'd', and the result superimposed on the original image is labelled 'e'. Figure 6.7 is labelled in the same manner, but shows how the technique fails to

cope with unpredictable physiology, in this case excess gas in the colon.

Figure 6.5: An example of segmenting a lung using the combination of a threshold and a watershed function.

Figure 6.6: An example of segmenting a lung using the combination of a threshold and a watershed function.

Figure 6.7: An example of how unpredictable physiology can affect the lung segmentation algorithm.

Figure 6.8 shows how the lung segmentation algorithm combined with the TB detection. This clearly reduces the number of false positives in regions out side the lung areas. The overall processing time was significantly reduces as regions of the image that are out side the lung areas were not searched for TB. The overall processing time was reduced to approximately 50% of the time taken to perform a TB search through an entire image. Unfortunately any errors in the lung segmentation can cause false negative errors in the resulting TB detections.

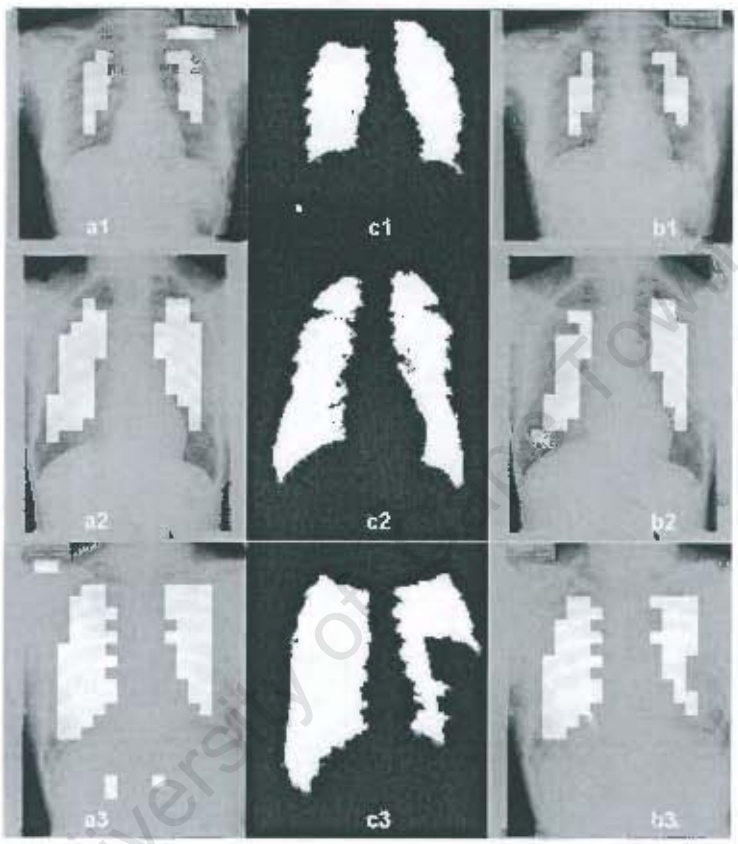

Figure 6.8: The combination of the lung segmentation with the TB detection algorithm. figures (*a#*) show the TB detection, (*b#*) shows the lung segmentation, and (*c#*) shows the combination of the two. Notice that errors in the lung segmentation cause false negatives in the combined images.

Chapter 7

Conclusions and Recommendations for Future Work

7.1 Conclusions

Image enhancement can be an extremely useful tool if it enhances important information in an image. It is possible, however, that the information enhanced is of no interest to the viewer in which case enhancing this information can distort the important detail.

Windowed edge detection was very effective at enhancing bone structure and even parts of the patients t-shirt. This could be extremely useful, especially if the radiologist is looking for bone fractures. However, in this project the primary concern is the analysis of lung tissue. This method of image enhancement distorted the texture formed by the pulmonary blood vessels and air passages in the lungs. As a result this process, although with interesting and possibly useful results, was not considered useful for aiding the diagnosis of miliary TB. Unsharp masking enhanced the lung textures very effectively and with considerably less image distortion than homomorphic image enhancement. Enhancement of the lung textures using wavelets seemed to have the best results when compared to edge detection, homomorphic enhancement and unsharp masking. The rolling ball algorithm was

very effective when used to remove the effects of ribs on the lung texture however it was extremely slow, which made it impractical to use on the large areas of lung texture in chest X-rays.

It is obviously impossible to enhance features in images that are not actually present. It is however possible to measure qualities in an image that cannot necessarily be measured by the human eye. Statistical methods can be used to accurately measure these sorts of qualities in images. Both the methods of granulometry and co-occurance matrices were unable to extract information that allowed for the successful classification of TB images verses healthy images. This is because these methods measure small sections of the structural element of the TB texture instead of the whole structural element. The natural lung textures are comprised of the same elements measured in the TB images making it impossible to distinguish between healthy and unhealthy lung textures.

Template matching simulates the human visual system ability to scan images looking for specific shapes. The Miliary TB texture in the lungs appears to be made up of many similarly sized dots, scattered evenly across the entire lung area. Creating an approximation to the shape of these balls, allows the computer to then scan through all the images in the data set looking for other similar shapes. Any areas with a large number of close matches should be examined more closely by a radiologist. Fourier domain correlation produced very promising results. The drawback being that the shape of the template being used was very general. This means that any diseases producing similar textures in the lungs will be classified as being miliary TB.

7.2 Recommendations for Future Work

Image enhancement is already a common tool for radiologists working with digitized X-ray images. Edge detection produced very effective results when enhancing bone structure, however this method does not seem to be used in clinical radiology. It is recommended that a further study into the effectiveness of finding fractures and other bone abnormalities using edge detection should be undertaken.

Statistical methods reviewed in this project did not produce significant results. It is possible that these methods, and other statistical methods, might produce significant results if the lung textures were preprocessed first. The rolling ball algorithm was very slow, but this algorithm might be able to improve the success of statistical methods.

Although the template matching was successful when distinguishing between healthy and unhealthy (TB) images, it will not be able to distinguish between other diseases that also cause granular textures in the lungs. The miliary TB can be seen as a very even texture throughout lungs in a chest X-ray. It is possible that measuring the proximity of the template matches will help distinguish between TB and some of the other lung diseases.

Other template shapes should be investigated to see what features can be detected. For example a 'top hat' template would be able to detect sharp edges in chest X-ray images. As X-ray images rarely have sharp edges, this could be used to detect errors in the image. This template might also be able to detect foreign objects, such as pieces of metal in chest X-rays.

The quality of the data set used in this project presented a degree of uncertainty in the results. It is recommended that for future studies in chest X-rays, a standard data set should be created. This would help compare results from one project with another. The images in the data set should be taken under controlled conditions to eliminate contrast variations between images.

The solution to the CAD of miliary TB does not have the ability to differentiate between miliary TB and other lung diseases like silicosis. It is recommended that further research into getting a computer to measure the differences in these textures should be undertaken.

In general it is considered highly unlikely that any CAD algorithm will be able to get 100% accurate results. It is recommended that image enhancement algorithms be used with (or independently of) such CAD systems to help the radiologist perform accurate diagnoses.

Appendix A

Pre-processing

This chapter describes some simple pre-processing tools that were repeatedly used during this project.

A.1 Thresholding

Thresholding can be used to divide a series of values into two classes. It can also be used to separate light and dark regions in images, resulting in a binary image [19, 6]. A binary image has only white and black pixels, which relate to the pixels that were either above or below the threshold value.

After a threshold has been assigned to a data set each value in the data set must then be compared to the threshold. All values above the threshold are assigned to one class and all those values below the threshold are assigned to the other class. The obvious problem is deciding on the value for the threshold. Choosing a threshold value depends on the data and the type of problem that needs to be solved. This section does not cover how to choose a threshold value. This is covered in the relevant chapters. This section covers some of the characteristics of using a threshold on an image.

Thresholding a noisy signal can result in multiple threshold crossings that make

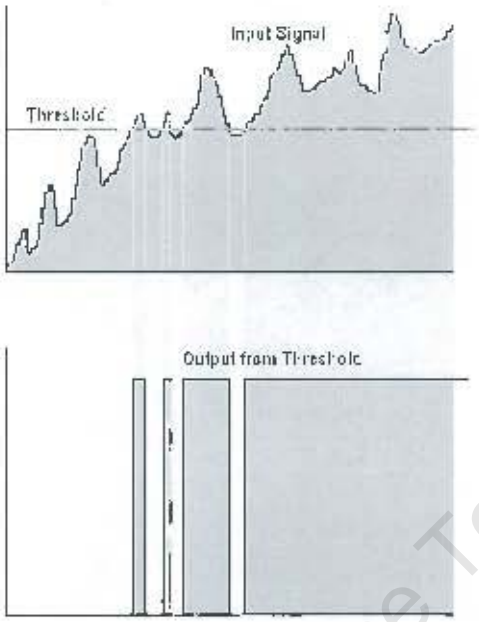

Figure A.1: Thresholding a noisy image results in brief crossings when the signal is near the value of the threshold.

the point at which the signal really crosses the threshold difficult to locate. This is demonstrated in Figure A.1. Notice that there are a number of points where the signal crosses the threshold briefly due to the noisy nature of the signal. Thresholding an image is therefore similar to thresholding an electric signal. As the signal approaches the threshold, noise in the signal tends to cause stuttering due to small spikes which cross the threshold. When applied to images this results in small islands of pixels scattered around the resulting binary image. Erosion, dilation and region labelling could be used to deal with these pixels.

As the intensity of each pixel is represented by a value, it is obvious that lighter regions can be separated from the darker regions with the use of an appropriate threshold. Figure A.3 was created by thresholding Figure A.2. Notice that, in the processed image, the edge of the skull was not clearly defined in all places, and that there were often a scattering of white and black pixels both inside and outside the skull. These were due to both noise and faint structures in the image.

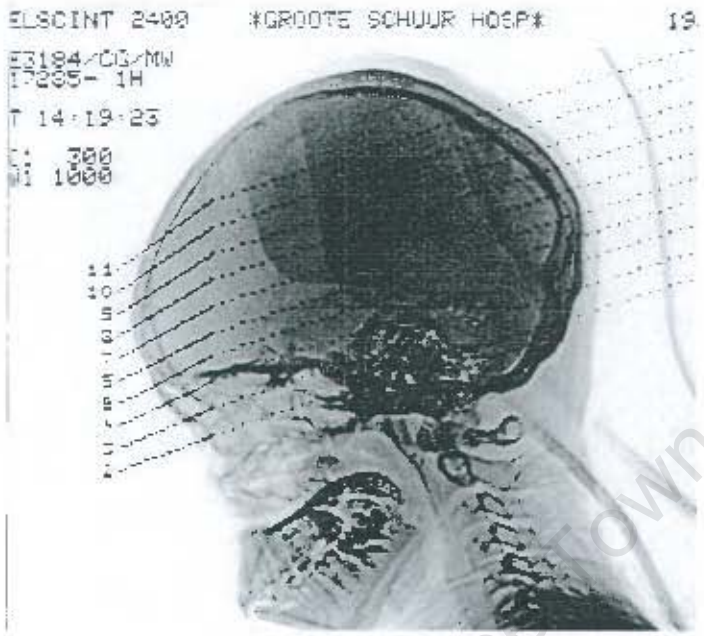

Figure A.2: An unprocessed X-ray image of a human head.

A.2 Erode and Dilate

If a binary image has a number of unwanted (or 'noisy') pixels scattered about the image, it is relatively easy to remove these pixels using either erosion, dilation or a combination of both [20]. The drawback is that the edges of the larger objects are altered slightly.

If a binary image has a large group of white pixels with a scattering of isolated white pixels around it, erosion can be used to remove all the isolated pixels. Erosion removes a layer of pixels from the outside of all groups of white pixels. The width of the layer removed is determined by the footprint size used. The footprint can be any size and shape, depending on the task.

Footprint here refers to a small binary image that is used as the region of effect of some functions. For example the footprint might be a white image with a size of 2 by 2 pixels, and the function could be erosion. Then a two pixel wide layer will

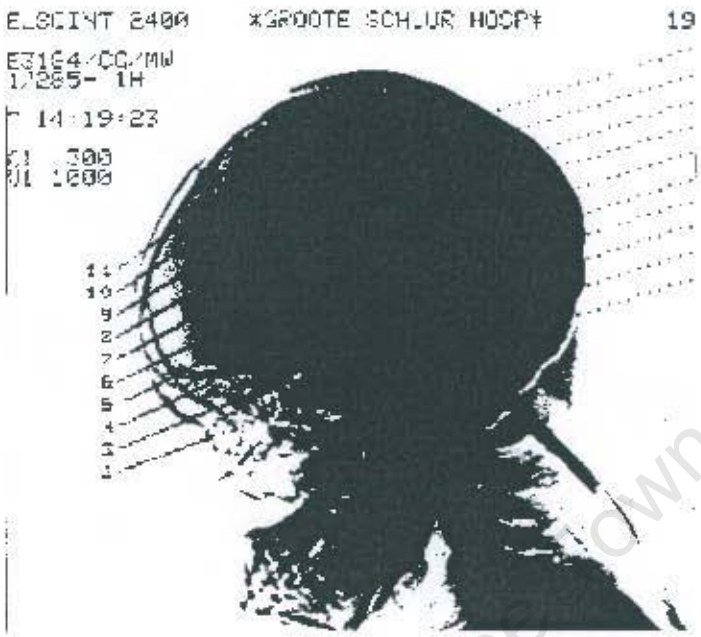

Figure A.3: A binary image, resulting from using a threshold on Figure A.2. Notice the scattering of white and black pixels near the edges of the skull.

be removed from the edges of all islands of white pixels in a binary image being processed.

When using the erode function, and pixel groups that have a radius less that the size of the footprint will be removed. Obviously larger objects will now be slightly smaller and their edges will have lost detail. Figure A.4 shows the effect of eroding figure A.3.

The inverse of this function is called dilation. Dilation adds a layer of white pixels to the outside of all groups of white pixels. But if there are any groups of black pixels surrounded by white, the black groups will be made smaller, or will disappear altogether. Figure A.5 shows the results of dilating figure A.3.

Using these two functions in conjunction will remove small groups of pixels, and keep the larger groups of pixels roughly the same size, although the edges of these

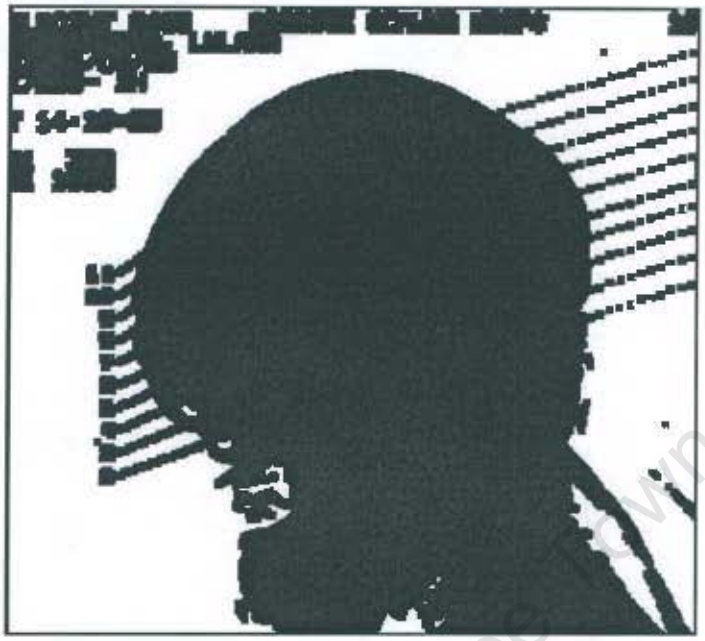

Figure A.4: The resulting image after eroding Figure A.3. Nearly all the small islands of white pixels in Figure A.3 have been removed, however all the black regions are now slightly larger

larger groups is then less detailed. Figure A.6 shows the effect of eroding figure A.5 which was the result of dilating figure A.3.

A.3 Labelling Regions

It is sometimes necessary to label regions or islands of pixels, typically in a binary image. This is done by assigning each region a unique colour. A region is any cluster of contiguous pixels with the same values. A colour histogram of this will then give the number of pixels in each region. This makes segmenting out regions of a certain size very easy [18, 4]. Figure A.8 is a region labelled version of Figure A.7, where each cluster was assigned an individual colour.

Figure A.5: The resulting image after dilating Figure A.3. Almost all the unwanted small regions of black pixels have been removed, for example, the text in the top left of Figure A.3 has been removed.

A.4 Histogram equalization

In all the images used in this project black was always 0. Brighter pixels, had higher values. White was always a power of 2 (ie. 2, 4, 8, 16 etc), depending on the bit depth of the image.

If an image is grey with no white or black pixels, it is relatively easy to stretch the histogram of the image to fit the full range from black to white [4, 21, 25]. This is implemented by subtracting the minimum pixel value from all the pixels. Every pixel is then divided by the maximum pixel value and multiplied by the value of white.

It is also possible to do a non-linear histogram stretch. This increases the contrast in one part of the histogram and decreases the contrast in another part. An example of this would be to square or to square root all the pixel values before doing a linear

Figure A.6: The resulting image after eroding the already dilated image in Figure A.5. Almost all the unwanted small regions of both black and white pixels have been removed. The larger regions are still roughly the same size as in the original, but their edges have been slightly altered.

Figure A.7: A number of unlabeled regions

histogram stretch.

Figure A.8: The Labelled regions from Figure A.7

In Figure A.9 all pixel values are relatively low, and the image therefore appears mostly dark. The histogram can be seen as the grey line in Figure A.12. Figure A.10 is the same image after its histogram had been linearly stretched. The new histogram can be seen as the black line in Figure A.12. The grey curve in Figure A.13 shows the same histogram after a non-linear stretch. In this example the non-linear stretch was done by square rooting the values. The black curve is the histogram of Figure A.2. Figure A.11 shows the resulting image from the non-linear stretch.

A histogram can be used to determine a threshold for segmenting out a region, with a different set of grey values, from the rest of the image. Dian [6] used this method to make an adaptive threshold for segmenting out the lungs in a full body chest X-ray from the LODOX machine.

A.5 Windowing

The human eye can only distinguish between aproximatly 40 different grey tones at any one time [26, 39]. Windowing is a good way to view small contrasts in images that can have thousands of different grey tones [25].

Windowing is very similar to histogram equalization, except it is used to look at a small portion of the histogram at a time. This is done by choosing an upper and

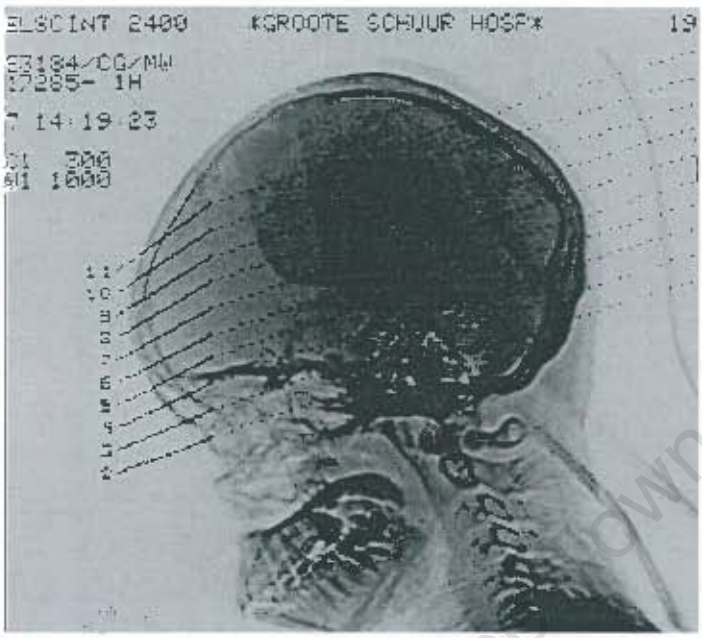

Figure A.9: An X-ray image of a human head, with low contrast.

lower limit for the pixel values in the image. All the pixels with values equal to or greater than the upper value are then set to the upper limit value. All values equal to or less than the lower value are set to the lower limit value. Then a standard histogram stretch is performed.

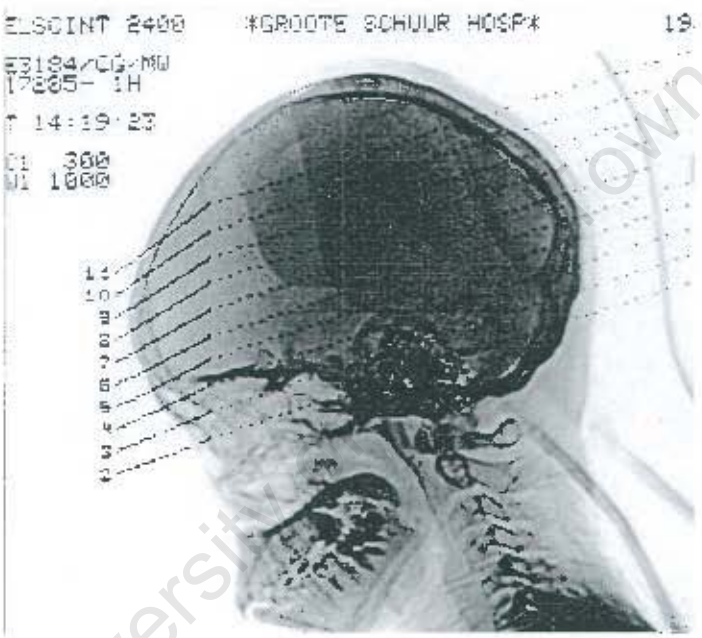

Figure A.10: Figure A.9 after its histogram had been linearly stretched.

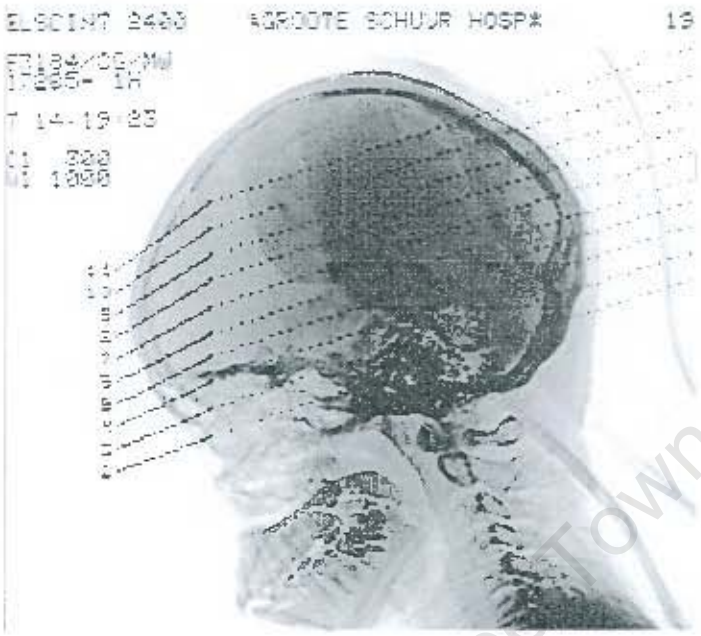

Figure A.11: The resulting image after a non-linear histogram stretch of Figure A.9

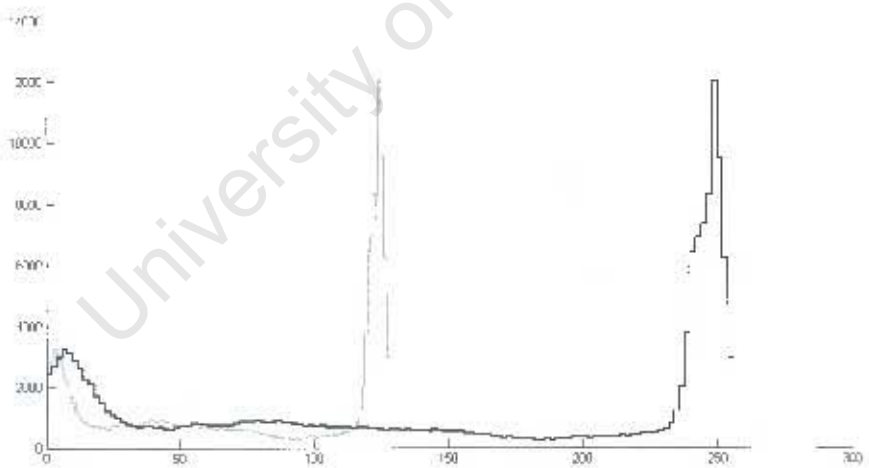

Figure A.12: The grey histogram is the distribution of pixel values from Figure A.9; the black histogram is a linearly stretched version of the grey histogram, resulting with the image in Figure A.2

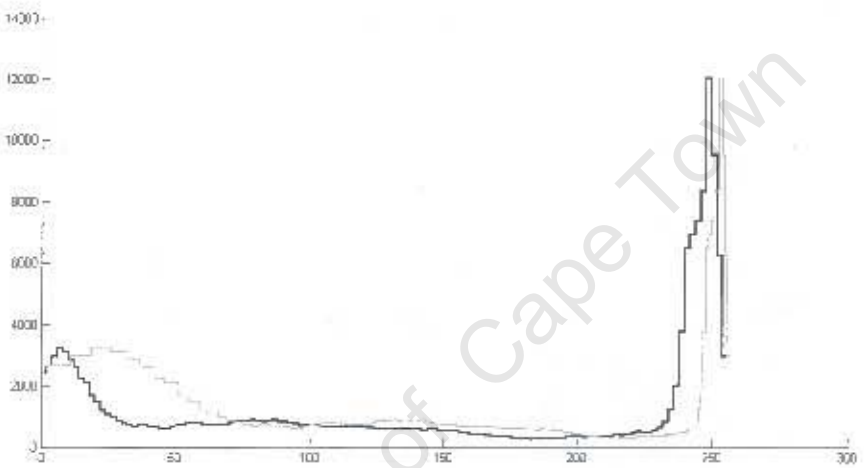

Figure A.13: The grey histogram is the distribution of pixel vales from Figure A.11; the black histogram is from Figure A.2. Notice that the histogram lengths (from lowest to the highest pixel values) are the same, but upper values of the black graph have been stretched at the expense of the lower values.

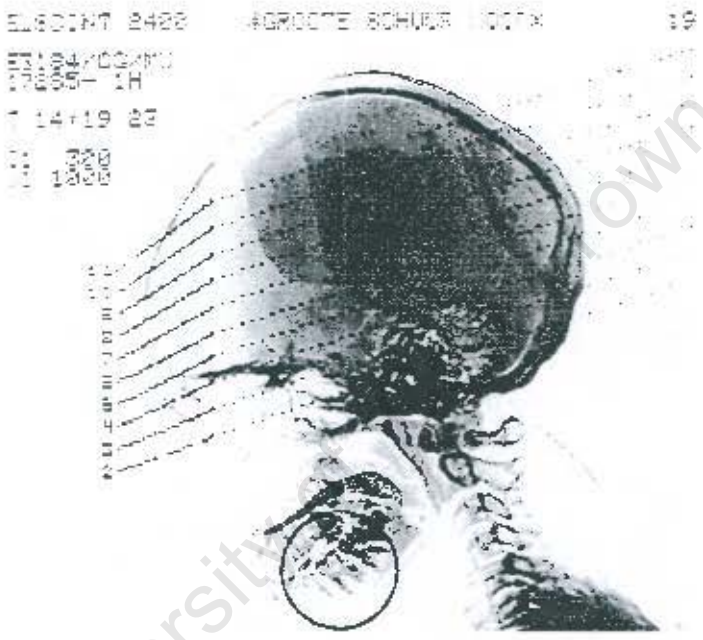

Figure A.14: Figure A.9 after windowing. Here one of the teeth (circled) has been made clearer at the expense of the rest of the detail in the image.

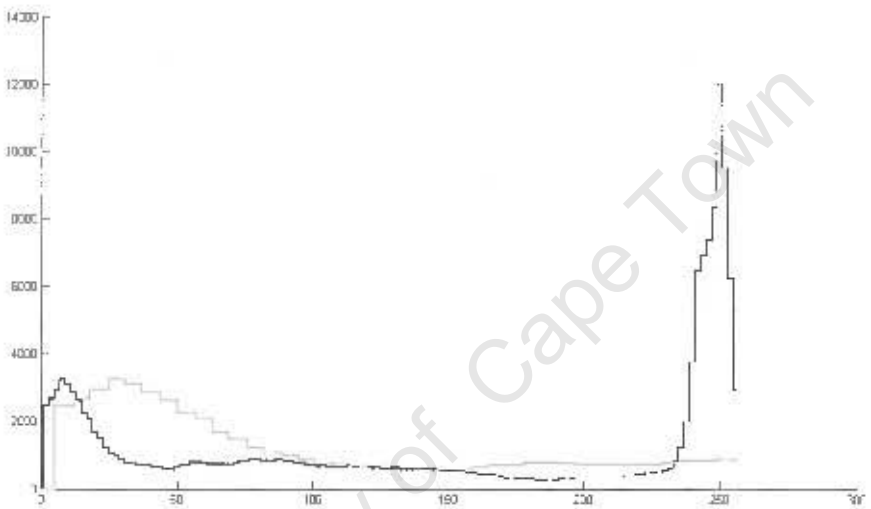

Figure A.15: The grey histogram is the distribution pixel values from Figure A.14; the black histogram is from Figure A.2. Notice that only a portion of the original black histogram is represented by the grey histogram, even though both extend over the whole range of pixel values from 0 to 255.

Appendix B

Receiver operating characteristic curve

Receiver operating characteristic curves (ROC) were originally used to test the radar operators during World War II [28]. Since then the medical profession has used ROC curves to interpret medical test results.

The ROC curve plots true positive rate versus false positive rate. Figure B.1 demonstrates an ROC, the black curve represents the results from a successful test. The grey diagonal line represents a completely random data set and so the further from this line the evaluated data set is the more accurate the classification.

The percentage of the area under the curve is used as a rough measure of this accuracy. The area is evaluated as in the following table.

100 – 90%	excellent
90 – 80%	good
80 – 70%	fair
70 – 60%	poor
60 – 50%	fail

Where possible, the results of tests in this project were plotted on a ROC curve.

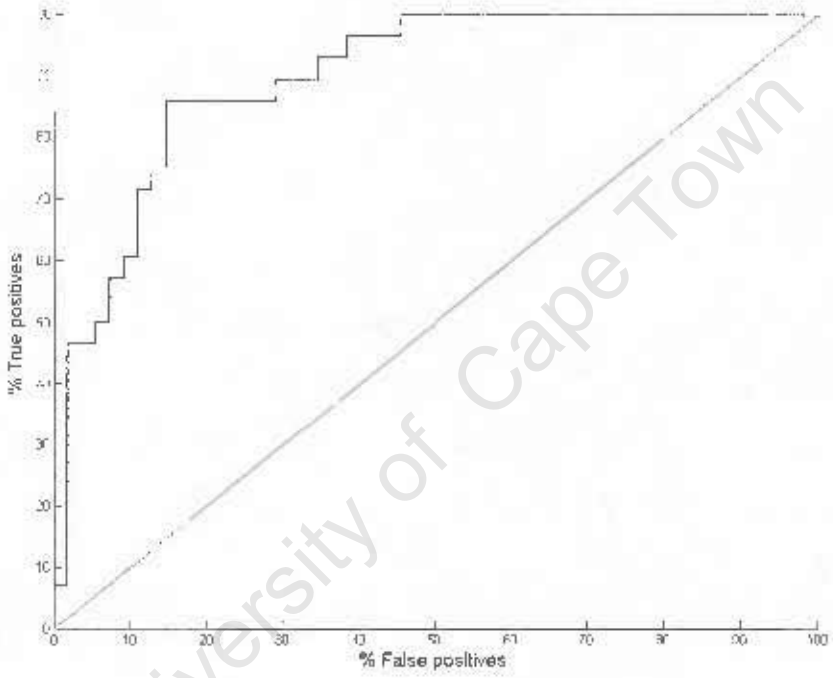

Figure B.1: An example of a receiver operating characteristic curve (ROC).

Appendix C

Images in the Fourier Domain

This Apendix covers some of the aspects of manipulating images in the Fourier domain.

The reference to high and low frequencies in an image is best illustrated in Figures C.1 and C.2. Figure C.1 has a high spatial frequency in the horizontal direction. This means that the contrast varies periodically, horizontally across the image. The periodicity is greater than the pattern in Figure C.2, and is therefore said to have a higher spatial frequency than the pattern in Figure C.2. This is easier to understand by looking at a horizontal row of pixels from each image. Figures C.3 and C.4 are plots of a single row of pixels from Figures C.1 and C.2 respectively. It is now easy to see that the pixel values vary sinusoidally. Converting these plots to the frequency domain and plotting the magnitude part of the frequency spectrum (Figures C.5 and C.6) shows that the high frequency pattern has a frequency component, (arrow in Figure C.5) further to the right than the low frequency pattern (arrow in Figure C.6).

These plots also show a second frequency component (indicated by an asterisk) on the far right of each plot. These components are the negative frequency components of the plot. The reason they are plotted on the far right, as opposed to being plotted to the left of the origin, is because computers don't work with matrices that have negative indices. This does not affect the Fourier domain methods used here, but it is nevertheless important to know where these components are. This

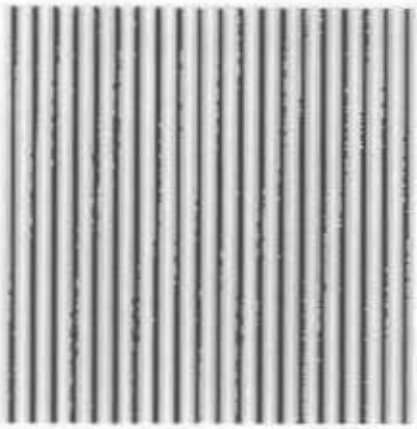

Figure C.1: A pattern showing a high horizontal spatial frequency

Figure C.2: A pattern showing a low horizontal spatial frequency

does mean that the high frequency components are located in the middle of the graph.

A simple example will serve to explain how certain spatial frequencies can be enhanced using the Fourier domain. Figure C.7 shows a simple high frequency signal superimposed on a large step function. The high frequency is the signal of interest, but because of its low amplitude it is barely noticeable in Figure C.7. The step function is made up largely of low frequency components. The filer shown in

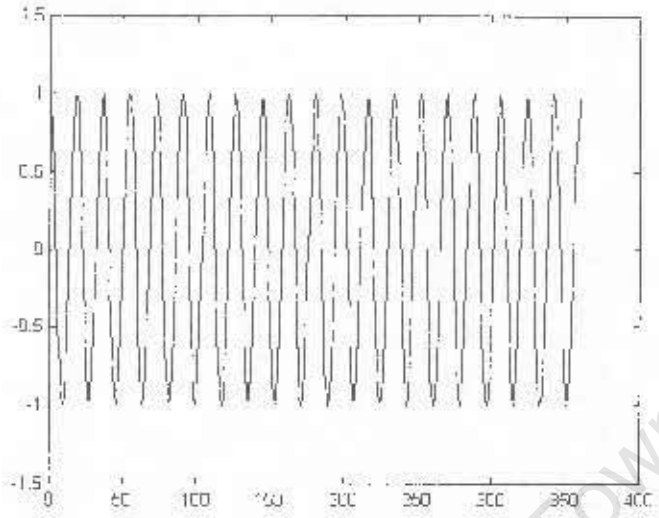

Figure C.3: The lightness intensities of a row of pixels from C.1 showing a high spatial frequency

Figure C.9 was designed to suppress low frequencies. This filter has low values for the low frequencies in the Fourier domain, and rises up to 1 for the remaining frequencies. The amount of suppression is governed by how low these values are made. The frequency range that is suppressed is controlled by the width of this part of the filter. Notice that this filter suppresses both the positive and negative low frequency components equally.

This filter was implemented by multiplying it with the magnitude components of the original signal. The product was then returned to the space domain, the result of which can be seen in Figure C.10. Here the original signal is plotted as a broken line and the filtered signal is a solid line. The filtered signal was scaled by 10 so that it could easily be compared to the original signal. This scaling factor was due to the filter dropping down to 0.1. Notice that the desired component of the signal was greatly enhanced, but the step function produced a significant amount of distortion. This distortion is always present, to vary degrees, and is the primary drawback to Fourier domain filtering.

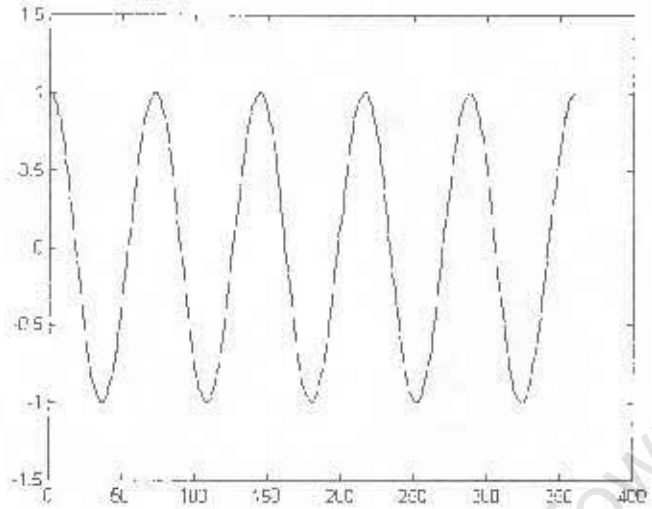

Figure C.4: The lightness of a row of pixels from Figure C.2 showing a low spatial frequency

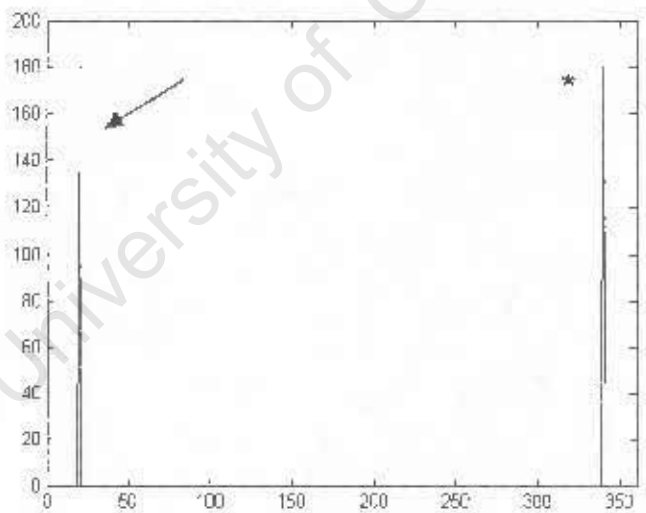

Figure C.5: The magnitude spectrum from Figure C.1

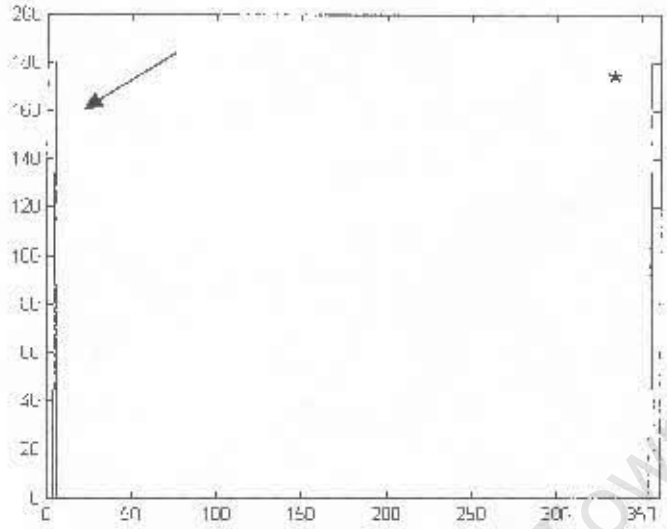

Figure C.6: The magnitude spectrum from Figure C.2

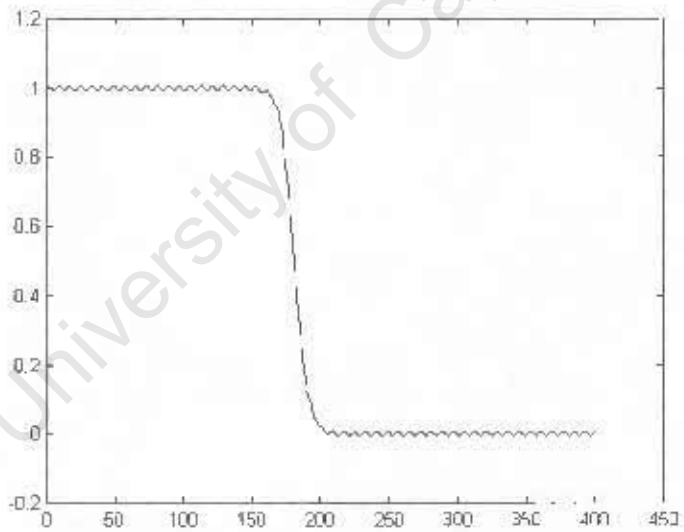

Figure C.7: A step function and a low amplitude high frequency signal.

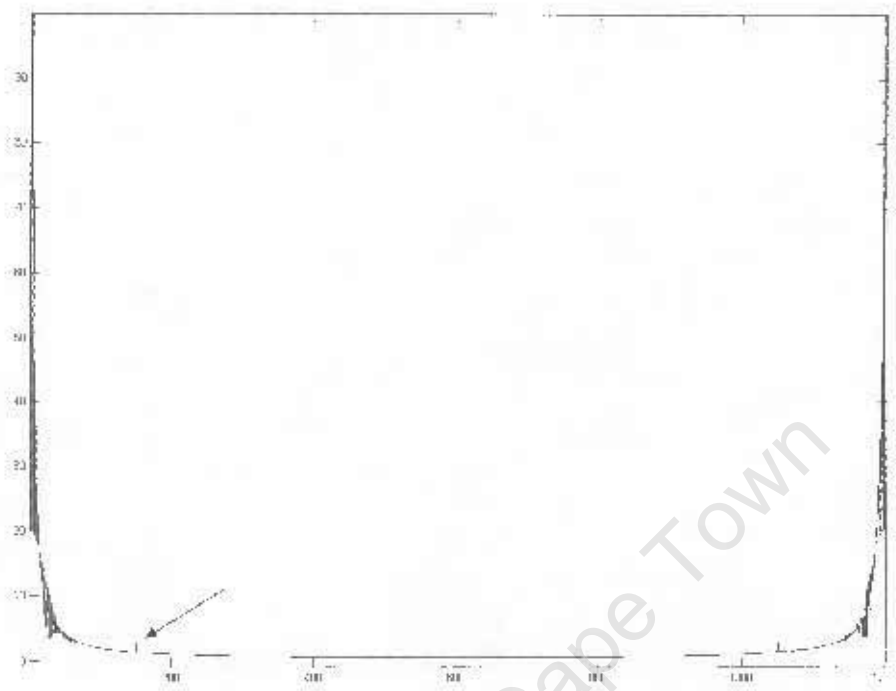

Figure C.8: The Fourier magnitude spectrum of Figure C.7.

In this simple example it is easy to adjust the filter so that this distortion is no longer visible. This was done by adjusting the filter so that it rose over a larger frequency range. The distortion can also be reduced by adjusting the level that the filter drops to. Making the shape of filter smoother will also reduces the distortion. The tradeoff is that the smoother the filter, the less the undesirable frequencies can be suppressed. This can be seen in Figures C.11 and C.12. The smoother rising filter is shown in Figure C.11, and its effect on the signal in Figure C.12. Note that the step function was preserved, but the high frequency signal (the information of interest) was substantially enhanced.

This method can be extended to two dimensions in order to filter the spatial frequencies in images. The Fourier domain components are very much the same as in the 2 dimensional example, with the low frequency components in the corners and higher frequency components at increasing radius's from the corners.

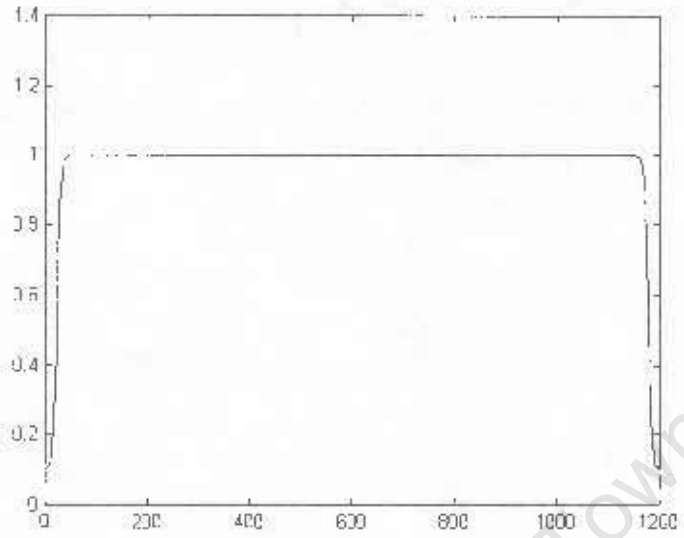

Figure C.9: A Fourier domain high pass filter

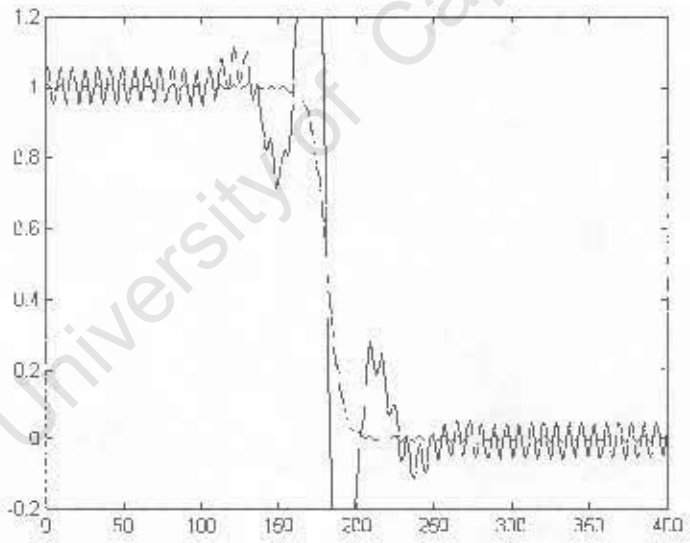

Figure C.10: The original signal from Figure C.7, shown as a broken line , and the result of using the filter shown in Figure C.9, as a solid line.

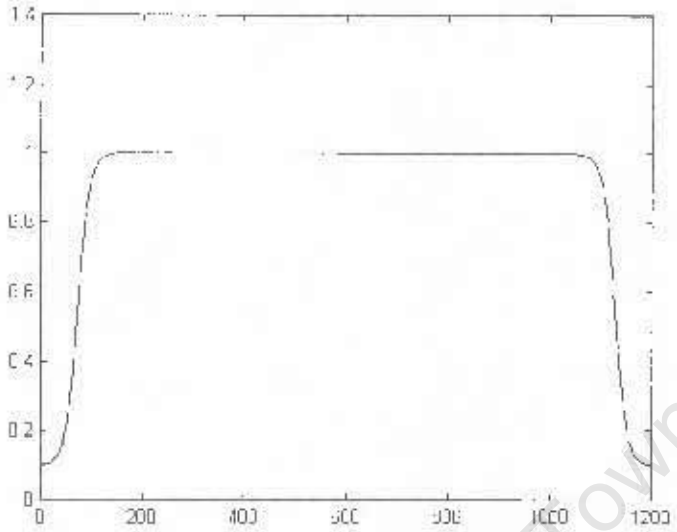

Figure C.11: An improved Fourier domain high pass filter

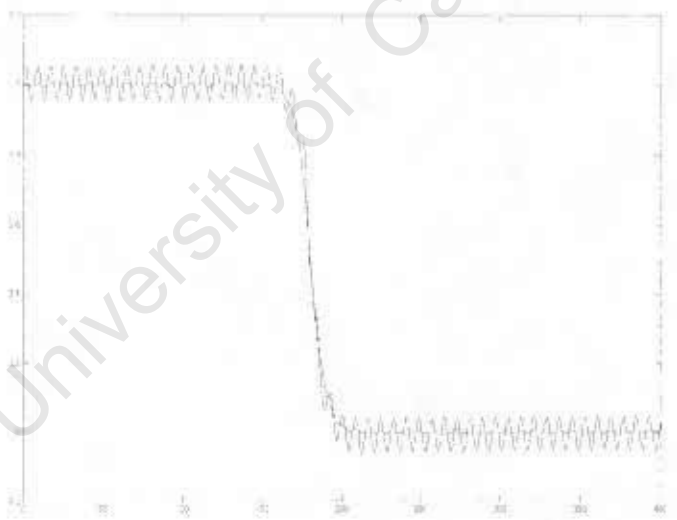

Figure C.12: The original signal from Figure C.7, shown as a broken line , and the result of using the filter from Figure C.11, shown as a solid line

Appendix D

Project Code

The next few pages are print outs of some of the code used in the implementation of this project.

```
function out = cooc(in)
%COOC returns the co occurance matrix for an image;
%The matrix 'in' must only have positive integers.
%The size of the output matrix depends on the max value of the matrix 'in'.

%Written by Anthony Koeslag while doing his Msc in image processing.
%17-06-2001

siz = size(in);

if min(min(in))<=0
    in = in-min(min(in))+1;
end

mx = max(max(in));
mn = min(min(in));

siz2 = mx;
out = zeros(siz2, siz2);

for x = 1:siz(1),
   for y = 1:siz(2)-1,
      if y<siz(2)
         xx = in(x,y);
         yy = in(x,y+1);
         out(xx,yy) = out(xx,yy) + 1;
         out(yy,xx) = out(yy,xx) + 1;
      end
   end
end

for x = 1:siz(1)-1,
   for y = 1:siz(2),
      if y<siz(2)
         xx = in(x,y);
         yy = in(x+1,y);
         out(xx,yy) = out(xx,yy) + 1;
         out(yy,xx) = out(yy,xx) + 1;
      end
   end
end

out = out;
```

```
function out = Lung_segmentation(im);

scl = 15;   %This is the amount that the lung image is scaled down by
            %the image is scaled to speed the process up, and nothing is realy
lost by doing this.

fprintf('Reducing size\n');
im_reduced = reduce(im, scl);   %reduce image here
clear im;
im_thresholded = im_reduced;
fprintf('Finding Edges\n');
im_edge = round(scale(abs(edge2(im_reduced)), 255));
im_reduced = round(scale(abs((im_reduced)), 255));
im_watershed_points = im_reduced.*0;

pause(0.001);
siz = size(im_watershed_points);

fprintf('Creating points for watershed\n');     %this next section creates
starting points for the watershed
dx = round(siz(1)/5);
dy = round(siz(2)/5);

tmp = im_reduced(2:dx, 2:dy); %top left
[m, iy] = min(min(tmp));
[m, ix] = min(tmp);
im_watershed_points(ix(iy)+1, iy+1) = 1;

tmp = im_reduced(2:dx, siz(2)-dy:siz(2)); %bottom left
[m, iy] = min(min(tmp));
[m, ix] = min(tmp);
im_watershed_points(ix(iy), iy+siz(2)-dy-1) = 1;

tmp = im_reduced(siz(1)-dx:siz(1), 2:dy); % top right
[m, iy] = min(min(tmp));
[m, ix] = min(tmp);
im_watershed_points(ix(iy)+siz(1)-dx-1, iy+1) = 1;

tmp = im_reduced(siz(1)-dx:siz(1), siz(2)-dy:siz(2)); %bottom right
[m, iy] = min(min(tmp));
[m, ix] = min(tmp);
im_watershed_points(ix(iy)+siz(1)-dx-1, iy+siz(2)-dy-1) = 1;

tmp = im_reduced(round(siz(1)/2-dx/2):round(siz(1)/2+dx/2), round(siz(2)*3/4-
dy/2):round(siz(2)*3/4+dy/2)); %middle right (Lung)
[m, iy] = min(min(tmp));
[m, ix] = min(tmp);
im_watershed_points(ix(iy)+round(siz(1)/2-dx/2), iy+round(siz(2)*3/4-dy/2)) = 1;
11(1) = ix(iy)+round(siz(1)/2-dx/2);
11(2) = iy+round(siz(2)*3/4-dy/2);

tmp = im_reduced(round(siz(1)/2-dx/2):round(siz(1)/2+dx/2), round(siz(2)/4-
dy/2):round(siz(2)/4+dy/2)); %middle left (Lung)
[m, iy] = min(min(tmp));
[m, ix] = min(tmp);
im_watershed_points(ix(iy)+round(siz(1)/2-dx/2), iy+round(siz(2)/4-dy/2)) = 1;
```

```
l2(1) = ix(iy)+round(siz(1)/2-dx/2);
l2(2) = iy+round(siz(2)/4-dy/2);

tmp = im_reduced(2:dx, round(siz(2)/2-dy/2):round(siz(2)/2+dy/2)); %top middle
[m, iy] = max(max(tmp));
[m, ix] =max(tmp);
im_watershed_points(ix(iy)+1, iy+round(siz(2)/2-dy/2)) = 1;

tmp = im_reduced(siz(1)-dx:siz(1), round(siz(2)/2-dy/2):round(siz(2)/2+dy/2));
%Bottom middle
[m, iy] = max(max(tmp));
[m, ix] = max(tmp);
im_watershed_points(ix(iy)+siz(1)-dx-1, iy+round(siz(2)/2-dy/2)) = 1;

tmp = im_reduced(round(siz(1)/4-dx/2):round(siz(1)/4+dx/2), round(siz(2)/2-
dy):round(siz(2)/2-1)); %top left of lung
[m, iy] = min(min(tmp));
[m, ix] = min(tmp);
im_watershed_points(ix(iy)+round(siz(1)/4-dx/2), iy+round(siz(2)/2-dy)) = 1;
l3(1) = ix(iy)+round(siz(1)/4-dx/2);
l3(2) = iy+round(siz(2)/2-dy);

tmp = im_reduced(round(siz(1)/4-dx/2):round(siz(1)/4+dx/2),
round(siz(2)/2):round(siz(2)/2+dy)); %top right of lung
[m, iy] = min(min(tmp));
[m, ix] = min(tmp);
im_watershed_points(ix(iy)+round(siz(1)/4-dx/2), iy+round(siz(2)/2)) = 1;
l4(1) = ix(iy)+round(siz(1)/4-dx/2);
l4(2) = iy+round(siz(2)/2);

tmp = im_reduced(round(siz(1)*3/4-dx/2):round(siz(1)*3/4+dx/2),
round(siz(2)/2):round(siz(2)/2+1.5*dy)); %bottom right of lung
[m, iy] = min(min(tmp));
[m, ix] = min(tmp);
im_watershed_points(ix(iy)+round(siz(1)*3/4-dx/2), iy+round(siz(2)/2)) = 1;
l5(1) = ix(iy)+round(siz(1)*3/4-dx/2);
l5(2) = iy+round(siz(2)/2);

tmp = im_reduced(round(siz(1)*3/4-dx/2):round(siz(1)*3/4+dx/2), round(siz(2)/2-
1.5*dy):round(siz(2)/2-1)); %bottom left of lung
[m, iy] = min(min(tmp));
[m, ix] = min(tmp);
im_watershed_points(ix(iy)+round(siz(1)*3/4-dx/2), iy+round(siz(2)/2-1.5*dy)) =
1;
l6(1) = ix(iy)+round(siz(1)*3/4-dx/2);
l6(2) = iy+round(siz(2)/2-1.5*dy);

im_watershed_points(1, :) = 1;              %box around the image
im_watershed_points(siz(1), :) = 1;
im_watershed_points(:, 1) = 1;
im_watershed_points(:, siz(2)) = 1;

%join just joins up the points that define the lungs here (used to get over each
rib)
im_watershed_points = join(im_watershed_points, l3(1), l3(2), l2(1), l2(2));
%top right to middle right lung
```

```
im_watershed_points = join(im_watershed_points, l6(1), l6(2), l2(1), l2(2));
%bottom right to middle right lung
im_watershed_points = join(im_watershed_points, l4(1), l4(2), l1(1), l1(2));
%top left to middle left lung
im_watershed_points = join(im_watershed_points, l5(1), l5(2), l1(1), l1(2));
%bottom left to middle left lung

pause(0.001);

im_watershed_points = bwlabel(im_watershed_points);     %labels each starting
point with its own colour
siz = size(im_reduced);
im_thresholded = im_thresholded(1:siz(1), 1:siz(2));
im_watershed_points = im_watershed_points(1:siz(1), 1:siz(2));

fprintf('Performing watershed\n');

im_watershed = watershedmex(im_reduced, im_watershed_points); %Keith Forbes
wrote this function, I just used it as is

pause(0.001);       %this part just chooses the results from the watershed that
represent the lungs, and converts them to a binary image
im_watershed = ((im_watershed==im_watershed_points(l1(1), l1(2))) +
(im_watershed==im_watershed_points(l2(1), l2(2))) +
(im_watershed==im_watershed_points(l3(1), l3(2))) +
(im_watershed==im_watershed_points(l4(1), l4(2))) +
(im_watershed==im_watershed_points(l5(1), l5(2))) +
(im_watershed==im_watershed_points(l6(1), l6(2))));
im_watershed = scale(im_watershed, 1);

threshold_value = max(max(im_thresholded))/3;
n = max(max(bwlabel(im_thresholded<=threshold_value)));
nm = 3;

fprintf('Performing adaptive threshold\n');

%this section isn't very smart, but it works. First the threshold is worked up,
then it is worked down
while ((sum(sum(im_thresholded<=threshold_value))/(siz(1)*siz(2)) <= 0.7)&(n >
nm)&(threshold_value<=max(max(im_thresholded))))|(sum(sum(im_thresholded<=thresh
old_value))/(siz(1)*siz(2)) <= 0.1)
   threshold_value = threshold_value+1;
   n = max(max(bwlabel(im_thresholded<=threshold_value)));
end

if threshold_value >=max(max(im_thresholded));
   n = 0;
   while (n < nm)&(threshold_value>0)
      threshold_value = threshold_value-1;
      n = max(max(bwlabel(im_thresholded<=threshold_value)));
   end
end

threshold_value = threshold_value+1;

pause(0.001);
```

```
out = (im_watershed&(im_thresholded<=(threshold_value)));
```

```
function out = rolling_ball(in, ball_size)
%this function returns the surface created by the rolling ball algorithm
%this is a 'closing' using the 'ball_size' as the smallest element
%this function is very slow, you have been warned!

imname = sprintf(['bal', '%0.0f', '.dat'], ball_size);                    %load
the ball from a file (faster than generating each time
ball_disc = scale(load(imname), 1);
sizz = size(ball_disc);
ball_disc = ball_disc.*ball_size;
ball_size = sizz(1);                                                      %the
matrix of the ball is slightly bigger than 'ball_size'

in = double(in);
siz = size(in);
temp_array = zeros(siz(1)+2*ball_size, siz(2)+2*ball_size);
%temp_array is a temporary array
temp_array(ball_size:siz(1)+ball_size-1, ball_size:siz(2)+ball_size-1) = in;
in = temp_array;
clear temp_array;
%don't need temp_array any more

temp_array2 = in.*0;
%temp_array2 is a temporary array
in_size = size(in);

last_time_check = 424242424242;
%last_time_check is a timer variable, put an arb number in here

tic
for y = 1:in_size(2)-ball_size+1;                                         %this is
where the ball is fitted under the matrix (in)
   for x = 1:in_size(1)-ball_size+1;                                      %b, c, f are
all temporary arrays
      b = in(x:x+ball_size-1, y:y+ball_size-1);
      b = b.*(ball_disc>0);
      b = b+((b==0).*(max(max(in)))) - ball_disc;
      c = min(min(b));
      f = temp_array2(x:x+ball_size-1, y:y+ball_size-1);
      temp_array2(x:x+ball_size-1, y:y+ball_size-1) =
f.*(((c+ball_disc).*(ball_disc>0))<=f) +
(c+ball_disc).*(((c+ball_disc).*(ball_disc>0))>f);

   end

   recent_time_check = round(100*(y/(in_size(2)-ball_size+1)));
%timer variables used to produce a running time (just so people know the program
hasn't hung and can go have coffee, and biscuts, and cheese cake, and bannana
muffins, and chocolate, and cocktails, and toasted cheese and tomato on white
bread, and albatros (with a twist of leming))
   if recent_time_check ~= last_time_check
      tm = (100-(100*(y/(in_size(2)))))*((toc)/(100*(y/(in_size(2)-
ball_size+1))));
      hr = floor(tm/3600);
      mn = floor((tm-hr*3600)/60);
      sc = floor(tm-hr*3600-mn*60);
```

```
      fprintf('Finnish in:  %dh  %dm  %ds\n', hr, mn, sc);
   end
   last_time_check = recent_time_check;
end
fprintf('\n');

pause(0.001);
out = temp_array2(ball_size:siz(1)+ball_size-1, ball_size:siz(2)+ball_size-1);
%trimming edges of array
```

```
function out = wavelet()
%This program Loads in an image, and enhances detail using the wavlet transform.

%Written by Anthony Koeslag while doing his Msc in image processing.
%26-06-2000

clear all;
close all;

tic
                                                        %Set your wavlet here.
wv = 'sym4';
                                                        %Set your contrast
adjustments here.
vl = [0.9, 0.8, 0.7, 0.6];      %1 wv = 'sym4'
%vl = [0.4, 0.5, 0.6, 0.75];    %2 wv = 'sym4'
%vl = [0.95, 0.95, 0.95, 0.4];  %3 wv = 'sym4'
%vl = [0.75, 0.6, 0.5, 0.4];    %4 wv = 'sym16'
%vl = [1, 1, 0.5, 0.5];         %5 wv = 'sym16'

                                                        %Load the image here,
also selecting small section and rotating image.
[im, com] = loadviff2('pic3.dbr');
im = rot90(scale(im(500:1100, 700:1300), 65535));

figure(1);
imshow(scale(im, 1), 'notruesize');
pause(0.001);

                                                        %this is how the image
is brocken down into its wavlet components.
[ca1,ch1,cv1,cd1] = dwt2(im, wv);
[ca2,ch2,cv2,cd2] = dwt2(ca1, wv);
[ca3,ch3,cv3,cd3] = dwt2(ca2, wv);
[ca4,ch4,cv4,cd4] = dwt2(ca3, wv);

                                                        %aply the contrast
adjustment and the invers wavlet transform
ca4b=ca4.*vl(4);
ca3b = idwt2(ca4b, ch4, cv4, cd4, wv);

sizz = size(ca3);
                                                        %WET PAINT! The 'Ca3b'
is no long the same size as 'Ca'. This might not be the best solution.
ca3b = ca3b(1:sizz(1), 1:sizz(2));
ca3b=ca3b.*vl(3);
ca2b = idwt2(ca3b, ch3, cv3, cd3, wv);

sizz = size(ca2);
ca2b = ca2b(1:sizz(1), 1:sizz(2));
ca2b=ca2b.*vl(2);
ca1b = idwt2(ca2b, ch2, cv2, cd2, wv);

sizz = size(ca1);
ca1b = ca1b(1:sizz(1), 1:sizz(2));
ca1b=ca1b.*vl(1);
```

```
im2 = idwt2(ca1b, ch1, cv1, cd1.*0, wv);


figure(2);
imshow(scale(im2, 1), 'notruesize');

toc
```

```
function out = TB_search()

% Use: "mcc -m -B sgl TB_search" to compile this

tic;

log_name = (setstr([cd, '\log.txt']));
log_fid = fopen(log_name, 'w');
fprintf(log_fid,
'************************************************************************%s', [13,
10]);
fprintf(log_fid, 'LOG CREATED:    %d-%d-%d    %d:%d:%f%s', clock, [13, 10, 13,
10]);
fclose(log_fid);

fprintf('\n************************************************************************
*\n');
fprintf('* TB_search.exe was written for AMI by Anthony Koesalg 2002
*\n');
fprintf('* This program will load all files with extensions .dbr and .raw,
*\n');
fprintf('* process them and save tham as .tif files
*\n');
fprintf('************************************************************************\
n\n');

template_size = 9; %can be any number from 3 to 18. would not recogmend 3, 8
seems to work the best.
block_size = 100; %always prossessing 300x300 blocks, This is the step size.
threshold = get_thresholds3;
threshold = threshold(template_size);

list = dir; %Start of getting the path to look in
files = size(list);
files = files(1);
f = 0;

new_dir = '\images';
dir_file = 'TB_search.ini';
dir_file_length = size(dir_file);
dir_file_length = dir_file_length(2);

for l = 3:files,
    name = list(l).name;
    letters = size(name);
    letters = letters(2);
    if letters == dir_file_length;
        if (sum(name == dir_file)==dir_file_length)&(list(l).isdir==0),
            f = f + 1;
        end
    end
end

if f == 0,

    dir_path = cd;
```

```
    siz2 = size(new_dir);
    siz = size(dir_path);
    dir_path(siz(2)+1:siz(2)+siz2(2)) = new_dir;
    dir_path = char(dir_path);
    save(dir_file, 'dir_path');

    %Funny thing to make the file a REAL ascii text file (actualy is another way
round this, but its already done, so...)

    fid = fopen(dir_file, 'w');
    fwrite(fid, dir_path, 'char');
    fclose(fid);

    dos_com = sprintf(['md ', new_dir(2:siz2(2))]);
    dos(dos_com);

    fprintf('- <%s> was not found.\n', dir_file);
    fprintf('- This file contains the directry in which to search for images to
\n- process.\n\n');
    fprintf('- <%s> was created\n', dir_file);
    fprintf('- <%s> set as the default directory.\n', dir_path);
    fprintf('- You can edit the directory in the <%s> file\n\n', dir_file);
    log_fid = fopen(log_name, 'a+');
    fprintf(log_fid, '- <%s> was not found.%s', dir_file, [13, 10]);
    fprintf(log_fid, '- This file contains the directry in which to search for
images to %s- process.%s', [13, 10], [13, 10, 13, 10]);
    fprintf(log_fid, '- <%s> was created%s', dir_file, [13, 10]);
    fprintf(log_fid, '- <%s> set as the default directory%s', dir_path, [13,
10]);
    fprintf(log_fid, '- You can edit the directory in the <%s> file%s',
dir_file, [13, 10, 13, 10]);
    fclose(log_fid);
end

%finding what path to read the images from

fid = fopen(dir_file, 'r');
name = dir(dir_file);
dir_path = fread(fid, [name.bytes, 1], 'char');
dir_path = setstr(dir_path');
original_path = cd;
fclose(fid);

cd(dir_path);

list = dir; %Start of getting all image files
list2.name = 0;
list2.date = 0;
list2.bytes = 0;
list2.isdir = 0;
files = size(list);
files = files(1);
f = 0;

for 1 = 3:files,
    name = list(1).name;
```

```
    letters = size(name);
    letters = letters(2);
    if letters >= 4;
        if ((sum(name(letters-3:letters)=='.raw')==4)|(sum(name(letters-
3:letters)=='.Raw')==4)|(sum(name(letters-
3:letters)=='.RAW')==4))&(list(1).isdir==0),

            ff = 0;
            nametiff = name(1:letters-3);
            nametiff = sprintf([nametiff, 'tif']);
            ff = ffind(nametiff, list);
            if ff == 0
                f = f + 1;
                list2(f).name = list(1).name;
                list2(f).date = list(1).date;
                list2(f).bytes = list(1).bytes;
                list2(f).isdir = list(1).isdir;
            else
                fprintf('- <%s> already exists, skipping <%s>\n', nametiff,
name);
                log_fid = fopen(log_name, 'a+');
                fprintf(log_fid, '- <%s> already exists, skipping <%s>%s',
nametiff, name, [13, 10]);
                fclose(log_fid);
            end
        end
        if ((sum(name(letters-3:letters)=='.dbr')==4)|(sum(name(letters-
3:letters)=='.Dbr')==4)|(sum(name(letters-
3:letters)=='.DBR')==4))&(list(1).isdir==0),

            ff = 0;
            nametiff = name(1:letters-3);
            nametiff = sprintf([nametiff, 'tif']);
            ff = ffind(nametiff, list);
            if ff == 0
                f = f + 1;
                list2(f).name = list(1).name;
                list2(f).date = list(1).date;
                list2(f).bytes = list(1).bytes;
                list2(f).isdir = list(1).isdir;
            else
                fprintf('- <%s> already exists, skipping <%s>\n', nametiff,
name);
                log_fid = fopen(log_name, 'a+');
                fprintf(log_fid, '- <%s> already exists, skipping <%s>%s',
nametiff, name, [13, 10]);
                fclose(log_fid);
            end
        end
    end
end
files = f;

f = 0;
if (files>0)&(sum(list2(1).name==0)~=1)&(sum(list2(1).name~=0)~=0),
for l = 1:files,
```

```
    out = 0;
    name = list2(1).name;
    letters = size(name);
    letters = letters(2);
    if letters >= 4;
        if ((sum(name(letters-3:letters)=='.raw')==4)|(sum(name(letters-
3:letters)=='.Raw')==4)|(sum(name(letters-
3:letters)=='.RAW')==4))&(list2(1).isdir==0),
            f = f + 1;
            fprintf('loading    %s     %s     %6.3f MB\n', name, list2(1).date,
(list2(1).bytes)/1000000);
            log_fid = fopen(log_name, 'a+');
            fprintf(log_fid, 'loading    %s     %s     %6.3f MB%s', name,
list2(1).date, (list2(1).bytes)/1000000, [13, 10]);
            fclose(log_fid);
            im = loadraw2(name);
            out = 1;
        end
        if ((sum(name(letters-3:letters)=='.dbr')==4)|(sum(name(letters-
3:letters)=='.Dbr')==4)|(sum(name(letters-
3:letters)=='.DBR')==4))&(list2(1).isdir==0),
            f = f + 1;
            fprintf('loading    %s     %s     %6.3f MB\n', name, list2(1).date,
(list2(1).bytes)/1000000);
            log_fid = fopen(log_name, 'a+');
            fprintf(log_fid, 'loading    %s     %s     %6.3f MB%s', name,
list2(1).date, (list2(1).bytes)/1000000, [13, 10]);
            fclose(log_fid);
            im = loadviff2(name);
            out = 1;
        end

        %Processing files here

        if out == 1,
            im = template_match(log_name, im, template_size, threshold,
block_size, name, f, files);
        end

        %Finnish processing files here

        if out == 1,
            name = name(1:letters-3);
            name = sprintf([name, 'tif']);
            fprintf('saving     %s', name);
            log_fid = fopen(log_name, 'a+');
            fprintf(log_fid, 'saving     %s', name);
            fclose(log_fid);
            imwrite(uint8(scale(im, 256)), name, 'tif');
            list3 = dir;
            files3 = size(list3);
            files3 = files3(1);
            for f3 = 3:files3,
                sz = size(name);
                sz = sz(2);
                sz2 = size(list3(f3).name);
```

```
                sz2 = sz2(2);
                if sz2 == sz,
                    if sum(name == list3(f3).name)==sz
                        fprintf('     %s     %6.3f MB\n\n', list3(f3).date,
(list3(f3).bytes)/1000000);
                        log_fid = fopen(log_name, 'a+');
                        fprintf(log_fid, '     %s     %6.3f MB%s',
list3(f3).date, (list3(f3).bytes)/1000000, [13, 10, 13, 10]);
                        fclose(log_fid);
                    end
                end
            end
        end
    end
end
end

t = toc;
t1 = floor(t/3600);
t2 = floor((t-t1*3600)/60);
t3 = (t-t2*60-t1*3600);

if f==1,
    fprintf('Finished processing %d file. Time elapsed %1.0fh %1.0fm %1.1fs \n',
f, t1, t2, t3);
    log_fid = fopen(log_name, 'a+');
    fprintf(log_fid, 'Finished processing %d file. Time elapsed %1.0fh %1.0fm
%1.1fs %s', f, t1, t2, t3, [13, 10]);
    fclose(log_fid);
end
if f>1,
    fprintf('Finished processing %d files. Time elapsed %1.0fh %1.0fm %1.1fs
\n', f, t1, t2, t3);
    log_fid = fopen(log_name, 'a+');
    fprintf(log_fid, 'Finished processing %d files. Time elapsed %1.0fh %1.0fm
%1.1fs %s', f, t1, t2, t3, [13, 10]);
    fclose(log_fid);
end

if f == 0
   fprintf('- No files with the extensions .dbr or .raw were found.\n');
   log_fid = fopen(log_name, 'a+');
   fprintf(log_fid, '- No files with the extensions .dbr or .raw were found.%s',
[13, 10]);
   fclose(log_fid);
end

log_fid = fopen(log_name, 'a+');
fprintf(log_fid, '- SESSION FINNISHED%s', [13, 10, 13, 10, 13, 10, 13, 10]);
fclose(log_fid);

cd(original_path);

out = f;
fclose('all');
pause;
```

```
function out = template_match(log_name, im, template_size, threshold,
block_size, name, f, f2);

out = 0;
dsc = parab_disc2(250, template_size);
dsc = dsc - mean(mean(dsc));
siz = size(im);
blk = 300;          %must always be 300 for the current thresholds!
blk_pad = 50+template_size;

dx = floor((siz(1)-(2*blk_pad))/block_size); %creating blocks to process
dy = floor((siz(2)-(2*blk_pad))/block_size);

while (blk_pad+dy*block_size+blk+blk_pad)>siz(2),
    dy = dy - 1;
end

while (blk_pad+dx*block_size+blk+blk_pad)>siz(1),
    dx = dx - 1;
end

pd = 1;

message = sprintf(['processing ', name, ' in progress. %0.0f of %0.0f files.'],
f, f2);

if ((f-1) >= 1)&(f~=f2)
t = toc;
tt = (t/(f-1))*(f2-f-1);
tt1 = floor(tt/3600);
tt2 = floor((tt - tt1*3600)/60);
tt3 = tt - tt1*3600 - tt2 *60;
message = sprintf(['processing ', name, ' in progress. %0.0f of %0.0f files.
ETA: %0.0fh %0.0fm %0.0fs'], f, f2, tt1, tt2, tt3);
end

sizm = size(message);
sizm = sizm(2);

out = scale(im, 1);

for y = 1:dy,
    for x = 1:dx,
        poi = im((x-1)*block_size+1:(x-1)*block_size+blk_pad*2+blk, (y-
1)*block_size+1:(y-1)*block_size+blk_pad*2+blk); %Part Of Image

        %correlate here;

        siz_p = size(poi);
        siz_d = size(dsc);
        dsc2 = corr2d(dsc, dsc);
        scl = max(max(dsc2));

        poi2 = abs(corr2d(scale(poi, 1), dsc));
        poi2(1:siz_d(1)*2, 1:siz_d(2)*2);
        poi2 = poi2(siz_d(1)*2+1:siz_p(1), siz_d(2)*2+1:siz_p(2))./scl;
```

```
        poi_mean = mean(mean(poi2));
        poi = scale(poi, 1);

        %finnish correlate here;

        dxy = size(poi);
        dxy(1) = floor((dxy(1)-block_size)/2);
        dxy(2) = floor((dxy(2)-block_size)/2);

        if poi_mean >= threshold,
            x1 = floor((siz_p(1)-block_size)/2)+(block_size)*(x-1)+1;
            x2 = floor((siz_p(1)-block_size)/2)+(block_size)*(x);
            y1 = floor((siz_p(2)-block_size)/2)+(block_size)*(y-1)+1;
            y2 = floor((siz_p(2)-block_size)/2)+(block_size)*(y);
            out(x1:x2, y1:y2) = out(x1:x2, y1:y2) + 1;
        end

        while ((100*((y-1)*dx+x)/(dx*dy))>=(100*pd/sizm))&(pd<=sizm),
            if (message(pd)=='\')&(message(pd+1)=='n'),
                fprintf('\n');
                log_fid = fopen(log_name, 'a+');
                fprintf(log_fid, '%s', [13, 10]);
                fclose(log_fid);
                pd = pd+2;
            else
               fprintf('%s', message(pd));
               log_fid = fopen(log_name, 'a+');
               fprintf(log_fid, '%s', message(pd));
               fclose(log_fid);
               pd = pd+1;
            end
        end
    end
end

log_fid = fopen(log_name, 'a+');
fprintf(log_fid, '%s', [13, 10]);
fclose(log_fid);
fprintf('\n');
out = toc;
```

Bibliography

[1] Calaway A and Wilson R. Curve extraction in images using the multiresolution fourier transform. Technical report, University of Warwick, Coventry CV4 7AL, England, 1990.

[2] Henrichsen A. Image quality measure of optical imaging systems. Technical report, Department of Electrical Engineering, University of Cape Town, ahenric@dip.ee.uct.ac.za, June 1999.

[3] Okumura A, Suzuki J, Furukawa I, Ono S, and Ashihara T. Signal analysis and compression performance evaluation of pathological microscopic images. *IEEE Transactions on Medical Imaging*, 16(6):701–710, December 1997.

[4] Koeslag AR. Detection of pathology in the lateral ventricals of the brain. Technical report, Department of Electrical Engineering, University of Cape Town, akoeslag@dip.ee.uct.ac.za, October 1999.

[5] Oppenheim AV and Schafer RW. *Digital Signal Processing*. Prentice-Hall, Inc., Englewoods Cliffs, New Jersey, 1975. Pages 480-531.

[6] Biagio D. Knowledge based segmentation of full-body medical x-rays images. Technical report, Department of Electrical Engineering, University of Cape Town, 1996.

[7] Krupinski EA. The radiologist and automated image analysis. In *Abstracts of Presentations at the Research Workshop on Automated Medical Image Analysis*, page 8. University of Ballarat, 1998.

[8] Werblin FS. The control of sensitivity in the retina. *Scientific American*, 228(1):70–79, January 1973.

[9] Li H, Ray Liu KJ, and Lo S-CB. Fractal modeling and segmentation for the enhancement of microcalcifications in digital mammograms. *IEEE Transactions on Medical Imaging*, 16(6), December 1997.

[10] Shikata H, Kitaoka H, Keserci B, Sato Y, and Tamura S. Quantitative evaluation of the spatial distribution of vessels surrounding pulmonary nodules. In H. U. Lemke, M. W. Vannier, K. Inamura, A. G. Farman, and K. Doi, editors, *Proceedings of the 14th International Congress and Exhibition: Computer Assisted Radiology and Surgery*, pages 761–766, 2000.

[11] Encyclopaedia Britannica Inc. *Koch, Robert*, volume V. Helen Hemingway Benton, Chicago, 1974. Page 864.

[12] Encyclopaedia Britannica Inc. *Pasteur, Louis*, volume VII. Helen Hemingway Benton, Chicago, 1974. Pages 790-791.

[13] Encyclopaedia Britannica Inc. *Rontgen, Wilhelm Conrad*, volume VIII. Helen Hemingway Benton, Chicago, 1974. Page 663.

[14] Encyclopaedia Britannica Inc. *Tuberculosis*, volume X. Helen Hemingway Benton, Chicago, 1974. Page 166.

[15] Declerck J, Feldmar J, Goris ML, and Betting F. Automatic registration and alignment on a template of cardiac stress and rest reoriented spect images. *IEEE Transactions on Medical Imaging*, 16(6):727–737, December 1997.

[16] Suzuki J, Furukawa I, Ono S, Kitamura M, and Ando Y. Contrast mapping and evaluation for electronic x-ray images on crt display monitor. *IEEE Transactions on Medical Imaging*, 16(6):772–784, December 1997.

[17] Zhang J and Huang HK. Automatic background recognition and removal (abrr) in computer radiography images. *IEEE Transactions on Medical Imaging*, 16(6):762–771, December 1997.

[18] Russ JC. *The Image Processing Handbook*. IEEE Press, second edition edition, 1995. Pages 1-76.

[19] Russ JC. *The Image Processing Handbook.* IEEE Press, second edition edition, 1995. Pages 481-546.

[20] Russ JC. *The Image Processing Handbook.* IEEE Press, second edition edition, 1995. Pages 407-480.

[21] Russ JC. *The Image Processing Handbook.* IEEE Press, second edition edition, 1995. Pages 211-282.

[22] Russ JC. *The Image Processing Handbook.* IEEE Press, second edition edition, 1995. Pages 283-346.

[23] Reinhardt JM, D'Souza ND, and Hoffman EA. Accurate measurment of intrathoracic airways. *IEEE Transactions on Medical Imaging*, 16(6):820–827, December 1997.

[24] Vincent L. Fast granulometric methods for the extraction of global image information. In Fazekas Z, Cooper GRJ, and Aghdasi F, editors, *Proceedings of the Eleventh Annual Symposium of the Pattern Recognition Association of South Africa*, pages 119–133. University of the Witwatersrand, Johannesburg, South Africa, 2000.

[25] Giger M and MacMahon M. Image processing and computer-aided diagnosis. *Image and Information Management: Computer Systems for a Changing Health Care Environment*, 34(3):565–596, May 1996.

[26] Mahowald MA and Mead C. The silicon retina. *Scientific American*, 264(5):40–47, May 1991.

[27] Morrison N. *Introduction to Fourier Analysis.* John Wiley & Sons, Inc., New Youk, Chichester, Brisbane,Toronto, Singapore, 1994. Pages 67-96.

[28] Tape TG. University of Nebraska Medical Center. Plotting and intrepretating an roc curve. Internet: http://gim.unmc.edu/dxtests/roc2.htm. Last accessed: 05-07-2002.

[29] Scott P. Symmetery detection in medical x-rays. Technical report, Department of Electrical Engineering, University of Cape Town, 1996.

[30] Davies PDO, Girling DJ, and Grange JM. *Oxford Textbook of Medicine*, volume 1, chapter Tuberculosis and its Problems in Developing Countries, pages 638–661. Oxford University Press, Oxford, 3rd edition, 1996.

[31] Bernardini R and Kovacevic J. Designing local orthoganal bases for evaluating image quality. Technical report, Signal Processing Research Department, AT&T Bell Laboratoties, 600 Mountain Avenue, Murry Hill, NJ 07974; rbernard@research.att.com, jelena@research.att.com.

[32] Gordon R, Herman GT, and Johnson SA. Image reconstruction from projections. *Scientific American*, 233(4):56–71, October 1975.

[33] Poli R. Genetic programming for image analysis. Csrp-96-1, School of Computer Science, The University of Birmingham, R.Poli@cs.bham.ac.uk, January 1996.

[34] Poli R and Cagnoni S. Evolution of pseudo-colouring algorithms for image enhancement with interactive genetic programming. Csrp-97-5, School of Computer Science, The University of Birmingham, R.Poli@cs.bham.ac.uk; Cagnoni@asp.die.unifi.it, January 1997.

[35] Polikar R. Fundamental concepts & an overview of the wavelet theory. Internet: http://www.public.iastate.edu/,rpolikar/WAVELETS/WTpart1.html, 2000. Last accessed: 05-07-2002.

[36] Haralick RM. Statistical and structural approaches to texture. In *Proceedings of the IEEE*, volume 67, pages 786–804, May 1979.

[37] Dippel S, Stahl M, Wiemker R, and Blaffert T. Multiscale enhancement of digital radiographs: A comparison of laplacian pyramid and fastwavelet transform. In H. U. Lemke, M. W. Vannier, K. Inamura, A. G. Farman, and K. Doi, editors, *Proceedings of the 14th International Congress and Exhibition: Computer Assisted Radiology and Surgery*, pages 507–512, 2000.

[38] Michael U. Texture classification and segmentation using wavelet frames. In *IEEE Transactions on Image Processing*, volume 4, pages 1549–1560, November 1995.

[39] van Ginneken B. *Computer-Aided Diagnosis in Chest Radiography*. PhD thesis, University Medical Center Utrecht, 2001. Pages 9-12; 13-18.

[40] van Ginneken B, ter Haar Romeny BM, and Viergever MA. Computer aided diagnosis in chest radiography: A survey. In *IEEE transactions on Medical Imaging*, volume 20, pages 1228–1241, December 2001.